4-5-2018

Workflows For X-ray And Neutron Interferometry/Tomography As Applied To Additive Manufacturing

Jumao Yuan
Louisiana State University and Agricultural and Mechanical College, jyuan4@lsu.edu

WORKFLOWS FOR X-RAY AND NEUTRON INTERFEROMETRY/TOMOGRAPHY AS APPLIED TO ADDITIVE MANUFACTURING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Chemistry

by
Jumao Yuan
B.S. in Chemistry, Beijing Normal University, 2012
May 2018

Acknowledgments

First of all, I would like to thank Dr. Leslie Butler, my advisor in Chemistry Department. During my Ph.D. study in the past five years, he helped me a lot with my research and even my life. With his recommendation, I joined in LA-SIGMA (The Louisiana Alliance for Simulation-Guided Materials Application) as a research assistant from summer in 2013 to spring in 2016. Moreover, Dr. Butler sponsored me several travel opportunities. Meanwhile, I would like to thank my committee members Dr. Kenneth Lopata, Dr. Megan Macnaughta from Chemistry and Dr. Clinton S Willson from Civil Engineering. I have taken a course from Dr. Lopata in my first year and I learned how to use Linux and Python programming in his class, which is a great benefit for my future research.

Dr. Kyungmin Ham from CAMD (Center for Advanced Microstructures and Devices) at LSU helped me with interferometry/tomography experiment and writing for my publications. At CAMD, under Dr. Kyungmin's instructions, I have learned how to use neuron interferometry efficiently for my sample data. Additionally, Dr. Jinghua Ge from CCT (Center for Computation and Technology) at LSU was an expert in Python programming and 3D visualization. Dr. Jinghua tutored me how to use VisTrails, Avizo and Jupyter notebook for image reconstruction and 3D visualization.

I also appreciate the support from Micheal Marsh and Eric Yen at ORS (Object Research System) company, offering us a free non-commercial software Dragonfly for 3D visualization and binarization/segmentation; while, Remi Rampin and Juliana Freire from NYU (New York University) provided us VisTrails workflow for imaging analysis in my research. Last but not the least, I would like to thank my group members during my Ph.D. study, Mutairu Olatinwo, Adam Brooks, Omoefe Kio and Godfrey Mills. They assisted me with instrument manipulation and data processing. With their great help, I could get along with my research and publications.

Table of Contents

List of Figures

Nomenclature

AM - Additive Manufacturing

DPC - Differential Phase Contrast

ABS (imaging) - Absorption

ABS (polymer) - Acrylonitrile Butadiene Styrene

PLA - Polylactic Acid

DF - Dark-field

SIRT - Simultaneous Iterative Reconstruction Technique

SART - Simultaneous Algebraic Reconstruction Technique

SAS - Small Angle Scattering

NYU - New York University

ORS - Object Research System

CAMD - Center for Advanced Microstructures and Devices

CCT - Center for Computation and Technology

CT - Computed Tomography

FBP - Filter Back Projection

ART - Algebraic Reconstruction Technique

GUI - Graphical User Interface

FFT - Fast Fourier Transform

APS - Advanced Photon Source

IDE - Integrated Development Environment

NLM - Nonlinear Regression Model

CUNY - City University of New York

NIST - National Institute of Standards and Technology

NIH - National Institutes of Health

SLM - Sintered Laser Melting

HZB - Helmholtz Zentrum Berlin

PSI - Paul Scherrer Institute

SEM - Scanning Electron Microscope

EBSD - Electron Backscatter Diffraction

HPC - High Performance Computing

GPU - Graphics Processing Unit

PSI - Paul Scherrer Institue

CNR - Contrast to Noise Ratio

CCD - Charge Coupled Device

FDM - Fused Deposition Modeling

Abstract

Grating-based interferometry/tomography is being rapidly developed for non-destructive evaluation of additive manufacturing test articles. An application requiring an efficient workflow is extremely necessary for stress and fatigue testing samples.

At present, scientific workflows play an important role for computational experiments in additive manufacturing 3D printing and interferometry/tomography imaging analysis. A clear workflow template allows scientists to process experiments easier and faster. Workflow library grows, but to find an appropriate workflow for their task is challenging. In our research, there are mainly three portions in the workflow, interferometry analysis, image reconstruction and 3D visualization. Currently, the hierarchy of workflows in interferometry/tomography projects is Mathematica, TomoPy/ASTRA/Jupyter notebook, VisTrails and Dragonfly. In general, two methods of interferometry analysis have been used in the first portion of workflow, single-shot interferometry and stepped-grating interferometry. As for the second portion, with a Jupyter notebook, the reconstruction method 'Gridrec' in TomoPy and 'SIRT' (Simultaneous Iterative Reconstruction Technique) in ASTRA generated a powerful reconstruction volume for absorption projections and dark-field projections separately. For the last portion, Dragonfly developed by ORS (Object Research System) company is a 3D visualization software with powerful scripting capabilities implemented in Python macros. Meanwhile, the VisTrails workflow incorporated both interferometry analysis and image reconstruction portions into VisTrails modules. Workflows in VisTrails hide much of the complexity of Mathematica or Python programming from users. Instead, with a simple GUI, it is possible for users to make their interferometry/tomography workflows through VisTrails modules.

Especially, for DPC (differential phase contrast) images in grating-based interferometry/tomography, we addressed the phase unwrapping issue with the method of 2D integra-

tion through generating phase images. With the algorithm, we have demonstrated the 2D integrated phase images denote a clearer contrast than DPC images.

Chapter 1
Introduction

Interferometry data processing is complicated, but it is worth the time and effort to generate a satisfying result. Different from traditional X-ray tomography, the combination of attenuation, phase and scattering images in grating-based interferometry allows new insights into sample structures. At present, grating-based interferometry is performed at a number of synchrotron and neutron imaging facilities as of 2017. A commercial X-ray laboratory system is on the market and a clinical trial is underway. Herein, an efficient workflow going through experimental raw images to 3D visualization is essential and valuable for data processing. Nevertheless, it is not easy to find an appropriate workflow for all experimental data. In our research, we have demonstrated that the workflows of our Mathematica, Jupyter notebook, VisTrails and Dragonfly successfully and effectively incorporated a friendly GUI and user-defined modules/macros for interferometry/tomography imaging analysis. In our lab, we use Mathematica workflows for development of new analysis or the incorporation of new equipment into the experiment. One example of a Mathematica workflow describes optimization of chemical engineering simulations with an emphasis on detecting sensitive parameters or control points [1]. Jupyter notebook has already been employed to extract data from a geology database and performs analysis from a range of over 400 established programs and functions [2]. Significantly, TomoPy in Jupyter notebook was developed to address the needs for tomographic reconstruction in an instrument-independent manner, primarily at synchrotron beamlines but also for users of neutron beamlines and laboratory X-ray instruments. Typically, TomoPy incorporated ASTRA package as a robust reconstruction toolbox with the reconstruction method of SIRT for dark-field image reconstruction. As a result, after obtaining reconstructed volumes from

TomoPy/ASTRA/Jupyter notebook, Dragonfly and VisTrails provide a reliable tomography GUI for users without background of Python programming. The tomography GUI combines features of both workflow and visualization systems, which is the penultimate goal for our tomography projects.

In general, our goal is to explore material structures with imaging analysis. Here, the workflows allow image processing correctly and efficiently. As for insights into materials, it is possible to observe internal features in samples, such as pores, fractures, cracks etc.

1.1 References

[1] Navarro, A.K.W., Vassiliadis, V.S. (2014) "computer algebra systems coming of age: Dynamic simulation and optimization of dae systems in mathematica (tm)". Computers & Chemical Engineering **62**: 125–138.

[2] Tauxe, L., Shaar, R., Jonestrask, L., Swanson-Hysell, N.L., Minnett, R., Koppers, A.A.P., Constable, C.G., Jarboe, N., Gaastra, K., Fairchild, L. (2016) "pmagpy: Software package for paleomagnetic data analysis and a bridge to the magnetics information consortium (magic) database". Geochemistry Geophysics Geosystems **17**(6): 2450–2463.

Chapter 2
Imaging

2.1 X-ray Imaging

Originally, Wilhelm Roentgen discovered X-ray beam during his study on cathode rays in 1895. In the experiment, he accidentally found the screen covered with barium platinocyanide was illuminated by emission of rays from a covered discharge tube, two meters away from the screen [1]. In the mean time, he took a photography of his wife's fingers with a ring upon her third finger. A variable transparency was shown in the fingers, flesh, wedding ring and bones. In the research, Roentgen found the permeability of X-ray varies with the density and thickness of product by analyzing four species, Platinum, Lead, Zinc and Aluminum. As a result, when product density decreases, the transparency increases rapidly.

From then on, as a non-destructive inspection technique, X-ray computed tomography (CT) has grown significantly with the features of improved spatial resolution and increasing availability in biomedical laboratory CT systems. The X-ray CT enables highly accurate insights into polymer materials, additive manufacturing 3D printing and fiber architectures [2]. Through examining cross-sections, it is possible to reduce the damage or loss of materials.

2.2 Neutron Imaging

The wavelength of neutron beam lies near $1\mathring{A}$ (0.1 nanometer) while for X-ray, it varies in the range of $0.01 \sim 10$ nanometers. In this case, with higher energy, neutron tomography enables penetrating bulk matter without damage to samples. Pioneering contributions to the development of neutron imaging technique, neutron spectroscopy and neutron diffraction technique were implemented by physical scientists Bertram N. Brockhouse [3] and

Clifford G. Shull [4] respectively. The neutrons start oscillation in atoms when penetrating the samples and the change in the density of neutrons will be first analyzed in the crystal. Due to the high H/D (Height to Diameter) ratio, neutron image makes it possible for stronger contrasts and scattering than X-rays.

2.3 Absorption, Dark-field and DPC

Grating-based interferometry in our lab produces conventional attenuation images (absorption in X-ray) as well as other two modalities, dark-field (DF) image and differential phase contrast (DPC) image. It is well known as the BeerLambert law for absorption sample [5],

$$A = -log_{10}\frac{P}{P_0} = abc \tag{2.1}$$

where $\frac{P}{P_0}$ is equal to transmission $\frac{T}{T_0}$, the ratio of initial intensity and the intensity when passing through sample, A is absorbance or optical density, a is absorptivity, b is the length of beam and c is concentration of the sample. However, it is hard to get internal structure information via absorption/attenuation images due to its poor X-ray imaging contrast.

Dark-field image, formed by small-angle scattering (SAS) of neutron or X-ray sources is particularly sensitive to density fluctuation and structural variations, offering many opportunities for insights into sample structures [6]. In Pfeiffer's paper [7], the author mentioned dark-field signal is proportional to the ratio of the visibility of the intensity oscillation, which can be written as,

$$DF = \frac{\frac{V_{ref}}{V_{sample}}}{\frac{T_{ref}}{T_{sample}}} \tag{2.2}$$

where V_{ref} and V_{sample} indicate the visibility of reference image and sample image, while T_{ref} and T_{sample} represent transmission images. The visibility V is related to intensity signal I for each pixel (m, n),

$$V(m,n) = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{2.3}$$

The SAS signals in the dark-field image denote the strong sensitivity to density variations in the sample with the range of nanometer to micrometer in pixel size.

To overcome poor contrast in absorption and dark-field, several methods have been investigated in the last decade. Nevertheless, those methods are largely dependent on the experimental setup and beam sources, such as analyser techniques [8, 9, 10], interferometric [11, 12] and free-space propagation [13, 14, 15, 16]. In Pfeiffer's paper [17, 18] of DPC image study, the author came up with a grating-based DPC setup, which could be effectively utilized to quantitatively analyze phase images. The phase contrast signal is closely related to phase shift value ϕ. In general, DPC shows the difference of phase shift between sample image and reference image.

$$DPC = \phi_{sample} - \phi_{ref} \tag{2.4}$$

where $\phi(x, y) = \frac{d\Phi(x,y)}{dx}$, a simple integration along Z-axis. However, due to instrumental instability and sample shifting or rotation, phase wrapping is a severe restriction to advance DPC image analysis. In the future study, integrated DPC images, named as phase image, might be a solution of phase unwrapping.

2.4 References

[1] RÖNTGEN, W.C. (1896) On a new kind of rays. Science **3**(59): 227–231.

[2] Garcea, S., Wang, Y., Withers, P. (2017) X-ray computed tomography of polymer composites. Composites Science and Technology.

[3] Brockhouse, B.N. (October 1995) Slow neutron spectroscopy and the grand atlas of the physical world. Reviews of Modern Physics **67**: 735–751.

[4] Shull, C., Wollan, E., Marney, M.: Neutron Diffraction Studies. (Oct 1948).

[5] Swinehart, D.F. (1962) The beer-lambert law. Journal of Chemical Education **39**(7): 333.

[6] Schaff, F., Bachmann, A., Zens, A., Zaeh, M.F., Pfeiffer, F., Herzen, J. (2017) Grating-based x-ray dark-field computed tomography for the characterization of friction stir welds: A feasibility study. Materials Characterization **129**(Supplement C): 143 – 148.

[7] Pfeiffer, F., Bech, M., Bunk, O., Kraft, P., Eikenberry, E.F., Brnnimann, C., Grnzweig,

C., David, C. (2008) Hard-x-ray dark-field imaging using a grating interferometer. **7**: 134.

[8] Ingal, V.N., Beliaevskaya, E.A. (1995) X-ray plane-wave topography observation of the phase contrast from a non-crystalline object. Journal of Physics D: Applied Physics **28**(11): 2314.

[9] Davis, T.J., Gao, D., Gureyev, T.E., Stevenson, A.W., Wilkins, S.W. (1995) Phase-contrast imaging of weakly absorbing materials using hard x-rays. **373**: 595.

[10] Chapman, D., Thomlinson, W., Johnston, R.E., Washburn, D., Pisano, E., Gmr, N., Zhong, Z., Menk, R., Arfelli, F., Sayers, D. (1997) Diffraction enhanced x-ray imaging. Physics in Medicine & Biology **42**(11): 2015.

[11] Bonse, U., Hart, M. (08 1965) An xray interferometer with long separated interfering beam paths. **7**: 99.

[12] Momose, A., Takeda, T., Itai, Y., Hirano, K. (1996) Phasecontrast xray computed tomography for observing biological soft tissues. **2**: 473.

[13] A. Snigirev, I. Snigireva, V.K.S.K., Schelokov, I. (04 1995) On the possibilities of x-ray phase contrast microimaging by coherent high energy synchrotron radiation. **66**: 5486–5492.

[14] WILKINS, S., Gureyev, T., Gao, D., POGANY, A., Stevenson, A. (11 1996) Phase-contrast imaging using polychromatic hard x-rays. **384**: 335–338.

[15] Cloetens, P., Ludwig, W., Baruchel, J., van dyck, D., Van Landuyt, J., Guigay, J.P., Schlenker, M. (12 1999) Holotomography: Quantitative phase tomography with micrometer resolution using hard synchrotron radiation x rays. **75**: 2912 – 2914.

[16] Vgberg, W., Larsson, D.H., Li, M., Arner, A., Hertz, H.M. (2015) X-ray phase-contrast tomography for high-spatial-resolution zebrafish muscle imaging. **5**: 16625.

[17] Pfeiffer, F., Weitkamp, T., Bunk, O., David, C. (03 2006) Phase retrieval and differential phase-contrast imaging with low-brilliance x-ray sources. **2**.

[18] F. Pfeiffer, M. Bech, O.B.T.D.B.H. (05 2009) X-ray dark-field and phase-contrast imaging using a grating interferometer. **105**.

Chapter 3
Interferometry

3.1 Background

Grating-based interferometry has been developed rapidly and become more widely practical. As for the setup, interferometry uses gratings to shift the phase of the X-ray in order to produce higher resolution images. There are typically three gratings used labeled G0, G1, and G2 with G0 being closest to the X-ray source and G2 being closest to the detector. The G0 grating, or the source grating, insures that the X-ray field is coherent. Gratings G1 and G2, also called phase and analyzer grating respectively, are what produce the contrast in the image. The sample can be placed between G0 and G1 or G1 and G2. The motion of the sample on a linear path with the grating allow for changes in the resolution of sample image.

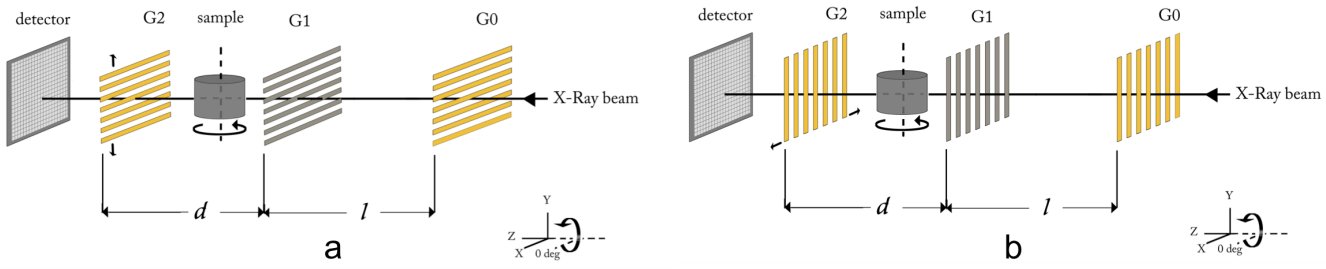


Figure 3.1: A Talbot-Lau stepped grating interferometer configured to observe dark-field images, i.e., small angle scattering (SAS), along two different laboratory axes (see triad axis). (a) The gratings are oriented for sensitivity along the laboratory Y-axis. (b) The gratings are oriented for sensitivity along the laboratory X-axis.

Due to the development of X-ray and neutron imaging facilities, a grating-based interferometry setup in commercial X-ray/neutron laboratories is demanded. For example, Zeiss [1], a optical and optoelectronic technology company in U.S. is aimed to design X-ray

[1] https://www.zeiss.com/corporate/us/home.html

spectroscopy for interferometry/tomography scientists. Meanwhile, Zeiss cooperated with Dragonfly software company [2] for interferometry/tomography 3D image analysis and visualization. Herein, after collecting images from experiment, data analysis is required to investigate sample structures and components. Here, two solutions for interferometry were proposed, stepped-grating interferometry analysis and single-shot interferometry analysis.

First of all, we create sinusoidal data as for a single pixel and process the data with a traditional nonlinear least squares package, an FFT (Fast Fourier Transform) analysis, and a linearized model of the sinusoid. The vectorized, linear model will be used in a later section for processing stepped-grating interferometry data. The nonlinear least squares method is slow, and the FFT is susceptible to noise from aperiodic sampling.

The equation for the synthetic data is

$$intensity = transmission + amplitude \times \sin(\theta + \phi) \tag{3.1}$$

Later, we will replace θ with $2\pi x_g/p_g$ where x_g is a translation distance for a grating and p_g is the period of that grating. ϕ here in the equation is phase shift value.

Gaussian noise is added with a standard deviation set to the square root of the intensity.

	θ,radians	sample	reference
1	0.698	1192	1798
2	1.396	1297	1800
3	2.094	1226	1666
4	2.793	1085	1480
5	3.491	880	1335
6	4.189	725	1226
7	4.887	778	1271
8	5.585	820	1483
9	6.283	985	1628
10	6.981	1153	1729
11	7.679	1267	1784
12	8.378	1235	1588
13	9.076	1075	1456
14	9.774	855	1326
15	10.472	734	1204
16	11.17	845	1258
17	11.868	881	1427
18	12.566	1027	1637

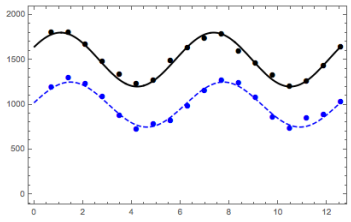


Figure 3.2: Synthetic data for a single pixel in a stepped-grating interferometer experiment. The two interferograms are for sample in the beam (blue) and sample out of the beam (black).

A standard procedure in data fitting is to use a non-linear model function such as $f(x) = a + b\sin(theta + c)$, equivalent to the equation that generated this model data, Eq. 3.1. Sometimes, constraints are imposed. In this fit, the constraints are $0 < b$ and

[2] http://www.theobjects.com/dragonfly/

$0 \leq x < \pi$. The reduced chi-square,

$$\chi^2_\nu = \left(\frac{1}{n-3-1}\right) \sum^{n} \frac{\left(y_i - y_i^{calc}\right)^2}{\sigma_i^2} \tag{3.2}$$

where σ_i is approximated by $\sqrt{y_i}$. We note that χ^2_ν is just over 1 indicating a good fit.

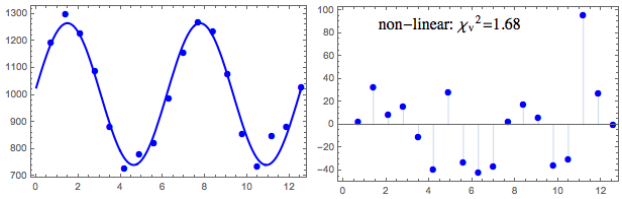


Figure 3.3: A non-linear least squares fit to the sample interferogram and residuals.

The fitted parameters are compared to the input parameters in this screenshot, Fig. 3.4.

	sample	reference
transmission	1000	1500
amplitude	250	300
phi	0.1	0.5

	Estimate	Standard Error	Confidence Interval
a	1505.29	7.0357	{1490.3, 1520.29}
b	280.421	9.94998	{259.213, 301.629}
c	0.476602	0.0354823	{0.400973, 0.55223}

	Estimate	Standard Error	Confidence Interval
a	1003.32	8.71864	{984.732, 1021.9}
b	262.63	12.33	{236.35, 288.911}
c	0.0923306	0.0469482	{−0.00773708, 0.192398}

Figure 3.4: (left) The input parameters to create the noise-free sinusoids. (right) The best fits for the parameters, the standard deviation, and the 95% confidence intervals for both the (top) reference pixel and (bottom) sample pixel.

The model function can be linearized as

$$f(x) = a + b\sin(\theta) + c\cos(\theta) \tag{3.3}$$

and then the three terms of $f(x)$ can be considered as basis vectors in a matrix solution.[1]

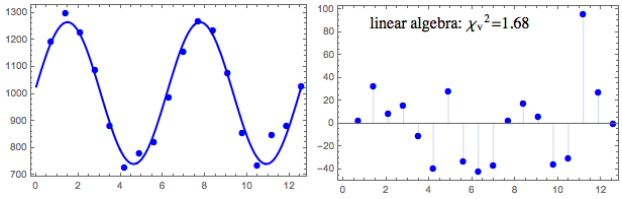


Figure 3.5: With the linearized model, Eq. 3.3, linear equation methods can be used to solve for the best fit.[1] Fortunately, the fit results are the same.

The sinusoidal data, when sampled periodically, lends itself to Fourier analysis. Grunzweig *et al.* [2] expressed the intensity as:

$$I(m,n,x_g) = \sum_{i=0} a_i(m,n)\cos\left[i2\pi x_g/p_g + \phi_i(m,n)\right] \tag{3.4}$$

$$\approx a_0(m,n) + a_1(m,n)\cos\left[2\pi x_g/p_g + \phi_i(m,n)\right] \tag{3.5}$$

where a_i and ϕ_i are the coefficients similar to those in Eq. 3.1 and (m,n) are coordinates in a pixelated image. The distances p_g and x_g refer to the grating period and the step-wise motion of the grating; the quantity $2\pi x_g/p_g$ is equivalent to θ.

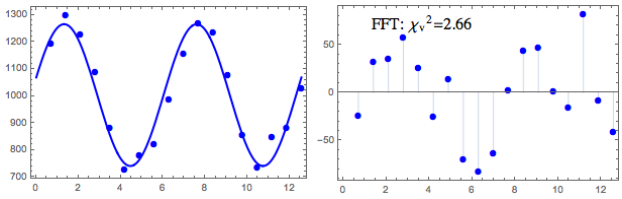


Figure 3.6: The synthetic data is fitted with FFT method.

The Mathematica code for the FFT method is listed below. Essentially, for a discrete FFT with zero-lag as the first element of a list, the 1-st and 3-rd elements of the list are needed to calculate the sinusoidal coefficients.

```
fitByFFT = Fourier[listDataSample, FourierParameters -> {0, 1}];
index = Reverse[Ordering[Abs[fitByFFT]]][[3]];
fftTransmission = Re[fitByFFT[[1]]] / Sqrt[Length[fitByFFT]];
fftAmplitude = 2*Abs[fitByFFT[[index]]] / Sqrt[Length[fitByFFT]];
fftPhi = ArcTan[Im[fitByFFT[[index]]], Re[fitByFFT[[index]]]]/Pi;
```

The `FourierParameters -> {0, 1}` ensures that the transform is performed with the discrete version of $\frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)\exp^{i\omega t}\,dt$.

The FFT method of interferogram analysis is sensitive to aperiodic sampling while the nonlinear least squares and the linear least squares are robust. The nonlinear least squares method is slow. Both the FFT method and the vectorized version of linear least squares

are fast. Vectorization of an image calculation is as simple as flattening the image into a 1D-vector, matrix multiplication, and the reshaping the result into image dimensions. The matrix reshaping and multiplication are fast and scale efficiently.

3.2 Stepped-grating Interferometry in Mathematica

For the scientist experienced with grating-based interferometry, here is a list of highlights discussed in this section:

- These results describe some of the highest spatial resolution grating interferometry performed anywhere. The high resolution brings out problems in the grating support structure in terms of noise in the absorption (Fig. 3.12), dark-field (Fig. 3.32), and differential phase contrast (Fig. 3.33) images. A proposed solution is post-acquisition alignment procedure to sub-pixel accuracy.

- Eq. 3.10 is a fast, robust solution to stepped-grating interferometry as well as for related problems such as K-edge tomography or any problem that can be expressed as a linear combination of basis vectors.

- A nonlinear least squares routine and the vectorized least squares algorithm return exactly the same coefficients, Figs. 3.8 and 3.9, respectively.

- Zooming into the background of the absorption image reveals a background with too much variation, Fig. 3.12, which is diminishing the image contrast-to-noise ratio.

- The best optimization metric for sub-pixel shift algorithms is the reduced chi-square image, Fig. 3.19.

The first experimental example is an interferogram of small biological sample, a foraminifera, a one-cell, ocean-dwelling protist. A raw image dimly shows the foram affixed to a wooden toothpick, Fig. 3.7. The striations in the background are attributed to support structures in the grating.

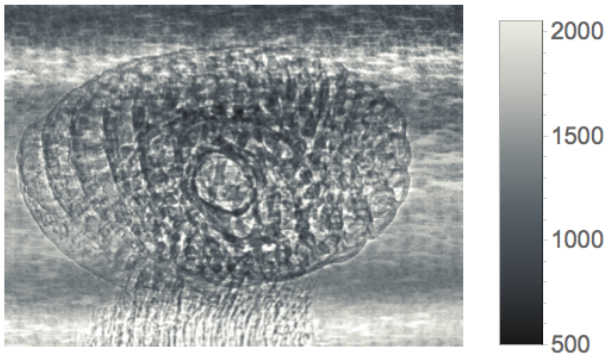


Figure 3.7: Raw image of the mostly calcium carbonate shell of a foraminifera, about 1 mm across. The colorbar gives the CCD counts. The gratings are aligned along the lab horizontal to take advantage of the vertical phase coherence of the beamline.

3.2.1 Theory of Vectorized Least Squares Analysis

The interferometry data are a set of X-ray counts c_{gp}, where $g = 1, ... M$ is the index that identifies the exposure, with one exposure at each grating displacement x_g; and $p = 1, ... N$ is the index that identifies the p-th pixel. Typically $M \sim 7$ to 16 and $N \sim$ millions. We have adapted a fitting algorithm from mathematical physics [1] that transforms the fitting operation into a simple matrix problem.

We fit each set of exposures pixel by pixel, c_{gp}, to each pixel's expected dependence on grating position x_g:

$$\hat{c}_{gp} = a_{1p} + a_p \sin\left(\frac{2\pi}{p_{grat}} x_g + \phi_p\right) \tag{3.6}$$

$$\equiv [1]\, a_{1p} + \left[\sin\left(\frac{2\pi}{p_{grat}} x_g\right)\right] a_p \cos(\phi_p) + \left[\cos\left(\frac{2\pi}{p_{grat}} x_g\right)\right] a_p \sin(\phi_p) \tag{3.7}$$

$$\equiv \sum_{\mu=1}^{3} B_{g\mu} a_{\mu p}$$

$$a_{2p} \equiv a_p \cos(\phi_p), \qquad a_{3p} \equiv a_p \sin(\phi_p), \tag{3.8}$$

where a_p and ϕ_p are the amplitude and phase of the interference term for the p-th pixel, and p_{grat} is the period of the translated grating. The interference term is expressed in both polar form (Eq. 3.6) and Cartesian form (Eq. 3.8). The latter is used to define the M×3

matrix **B** (Eq. 3.8) that represents the three fitting functions shown in brackets in Eq. 3.7: constant, sine, and cosine. The $a_{\mu p}$ is one element in the coefficient matrix **a**, which is a 3×N matrix of amplitudes, or weights, of the three fitting functions for the N pixels. Later, we will reshape **a** into a 3D matrix with dimensionality [rows, columns, 3].

The best fit can be chosen to be the set of coefficients $a_{\mu p}$ that minimize the deviation-squared, D_p, for each pixel, defined by

$$D_p \equiv \sum_{g=1}^{M} (c_{gp} - \hat{c}_{gp})^2 \ . \tag{3.9}$$

To minimize D_p, we need only calculate the derivatives of each deviation-squared with respect to each component of **a**, set all deviations to zero, and solve the resulting matrix equations. The closed form expression for the coefficient matrix **a** is found to be

$$\mathbf{a} = \mathbf{G} \cdot \mathbf{c} \ , \tag{3.10}$$

where

$$\mathbf{G} = (\mathbf{B}^T \cdot \mathbf{B})^{-1} \cdot \mathbf{B}^T \ , \tag{3.11}$$

and where the superscript T indicates the matrix transpose. So the optimization problem is reduced to multiplying a 3×M fixed matrix **G** into the M×N data matrix **c**, to find the 3×N coefficient matrix **a**. With an efficient matrix manipulation program like 'numpy' package in Python, the multiplication is extremely fast, on the order of one second for $1k \times 1k$ images times sixteen grating steps. By inspection, we recover the polar coefficients from the Cartesian in the usual way:

$$a_p = \sqrt{a_{2p}^2 + a_{3p}^2} \tag{3.12}$$

$$\phi_p = \tan^{-1}(a_{3p}/a_{2p}) \text{ (two quadrant)} \tag{3.13}$$

$$= \tan^{-1}(a_{2p}, a_{3p}) \text{ (four quadrant).} \tag{3.14}$$

The model equation, Eq. 3.8, can be extended to include anharmonic terms to describe the grating effects more accurately and to include more sophisticated grating motion trajectories.

The implementation of Eq. 3.8 in Mathematica is

```
b1 = Table[1,{i,numberGratingSteps}];
b2 = Table[Sin[2 \[Pi] listGratingStepsMicron[[i]]/gratingPeriodMicron]//N,
       {i,numberGratingSteps}];
b3 = Table[Cos[2 \[Pi] listGratingStepsMicron[[i]]/gratingPeriodMicron]//N,
       {i,numberGratingSteps}];
bVector = Transpose[{b1,b2,b3} ]];
```

yielding `bVector`, the M×3 **B** basis vectors. We again note that anharmonic terms can be added at this point by simply adding more terms to **B**. Also, **B** can account for missing grating positions, grating positions extending over more than one period, and aperiodic grating positions, simply by editing the elements of the list, `listGratingStepsMicron`. **B** is calculated once in an interferometry/tomography calculation.

The important 3×N matrix **G** is also calculated once, where Eq 3.11 is implemented as:

```
gMatrix = Inverse[Transpose[bVector] . bVector] . Transpose[bVector];
```

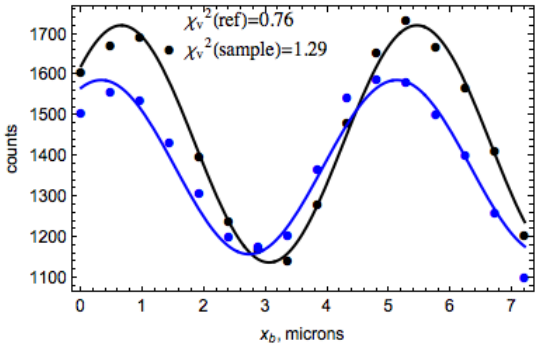
The calculation that is repeated in an interferometry/tomography calculation, and dominated by storage access times, is Eq. 3.10 as implemented in

```
allSampleData= ConstantArray[0,{rows,columns,numberGratingSteps}];
For[k=1,k<=numberGratingSteps,k++,
  oneImage = Import[filenamesSample[[k]],"TIFF"];
  oneData = ImageData[oneImage,"Bit16"]- dataDark;
  allSampleData[[All,All,k]]=oneData;  ];
cVector=Transpose[Flatten[allSampleData,{1,2}]];
```

```
aVector=gMatrix.cVector;
aMatrix=Partition[Transpose[aVector],columns];
```

where `aVector` is 3×N and matrix reshaping yields **a** (`aMatrix`) with dimensions `rows`×`columns`×3 where the last dimension contains the coefficients a_{1p}, a_{2p}, and a_{3p} for every p-th pixel. The sinusoidal signal is characterized by $transmission = a_{1p}$, $amplitude = a_p = \sqrt{a_{2p}^2 + a_{3p}^2}$, and $\phi = \tan^{-1}(a_{2p}, a_{3p})$. For ϕ, please note the different orderings of a_{2p} and a_{3p} in two-quadrant and four-quadrant arctan functions.

Fig. 3.8 shows an analysis of the data point at the center of the sample and reference images. The sample image for the first grating position is shown in Fig. 3.7 and the fits to the sample and reference interferograms are done with the linear algebra procedure discussed above.



	sample	reference
a_{1p}	1372.49	1429.58
a_{2p}	88.4182	219.55
a_{3p}	195.062	191.925

	sample	reference
transmission	1372.5	1429.6
amplitude	214.2	291.6
phi,radians	1.145	0.718

Figure 3.8: The data points were extracted from the center of the 16 sample images, including Fig. 3.7, and 16 reference images. The fits were made with Eq. 3.10.

As a validation of the linear algebra procedure and code, the same data points are fitted using a non-linear least squares model, Eq. 3.1, and shown in Fig. 3.9. The results are identical to the linear algebra procedure.

3.2.2 Experiment

A two-grating interferometer was assembled at the Advanced Photon Source 2-BM-B beamline.[3] The stepped-grating system was installed 23 m downstream of the double multilayer monochromator, which provides a monochromatic beam at 25 keV with 1% bandwidth. The phase grating, G_1, was optimized for π-phase shift at 25 keV with a period

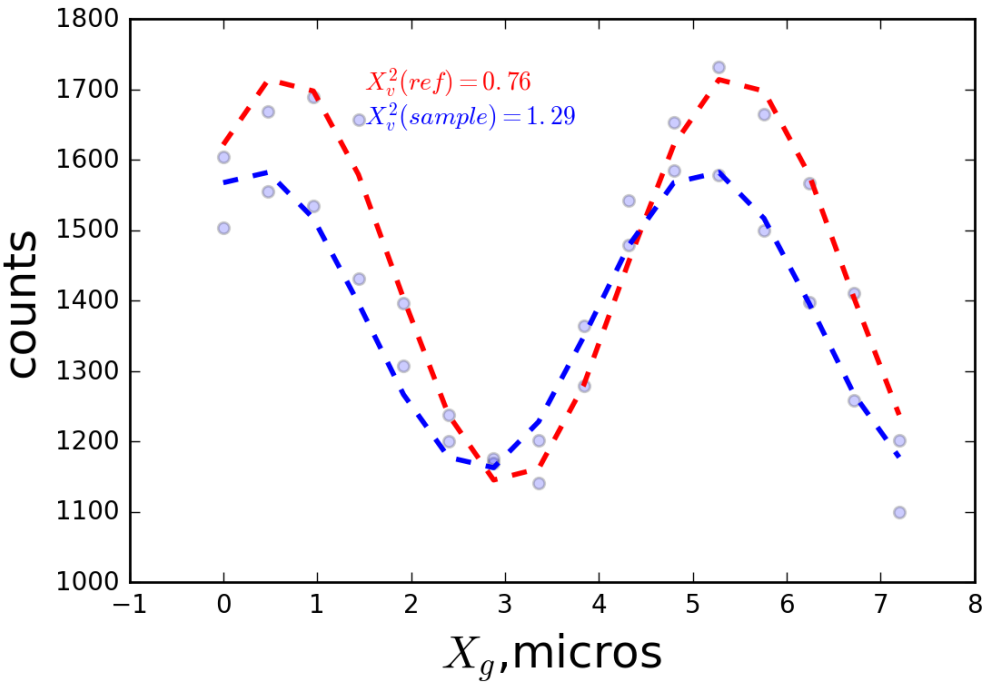


Figure 3.9: A non-linear least squares fit to Eq. 3.1. The results are identical to the linear algebra procedure and code used to prepare Fig. 3.8. The confidence intervals are at the 95% level. The top table gives the fit parameters for the reference interferogram (black) and the bottom table is for the sample (blue).

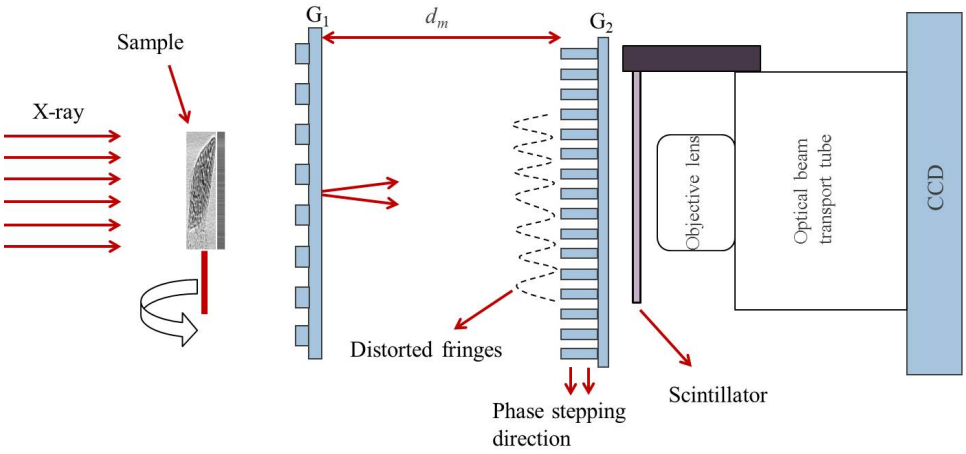


Figure 3.10: A schematic side-view of the experimental setup for the two-grating system. The first fractional Talbot distance, $m = 1$, occures at d_m=58 mm for grating periods of G_1=4.8 μm (phase grating) and G_2=2.4 μm (analyzer grating). The sample-to-detector distance is 270 mm.

of 4.8 μm; the analyzer grating, G_2, had a period of 2.4 μm. The gratings were produced by Microworks (Karlsruhe, Germany) and to support the high-aspect ratio structures, both

gratings included support structures: the G_1 phase grating was fabricated with a broken-line structure and the G_2 analyzer included a bridge-structure connecting adjacent lines.

Experiments were performed with a sample-to-phase grating distance of 270 mm; a much shorter distance was desired but not possible due to mechanical interference between stacks of positioning stages in the test setup. The analyzer grating was then positioned at the first fractional Talbot distance ($m = 1$) 58 mm from the phase grating. Both phase and analyzer gratings were mounted on dual-tilt stages. Due to a smaller source size along the vertical plane, the X-ray radiation has its greatest coherence along the lab vertical; both gratings were positioned with the gratings aligned with the lab horizontal. The tomography sample rotation was about the lab vertical axis. The stepping scan motion along the lab vertical axis was performed with a piezoelectric-based positioner stage with a 200-μm range and sub-nanometer resolution. A 100-μm thick LuAG:Ce scintillator was imaged with a 10× optical magnification lens and a Coolsnap HQ2 CCD with a 1040×1392 array of 6.45-μm-square pixels; the small effective pixel size of 0.645-μm was used to accommodate another experiment performed in the test setup. The exposure time for each interferogram was 600 ms, and 16 interferograms were measured across 4/3 period of the analyzer grating structure. Reference data were collected before and after the tomography data; 601 projections were acquired in the angle range of 0° to 180°. Also, background images (X-rays off) were collected to account for CCD dark-current and the offset signal.[4]

3.2.3 Transmission and Absorption Images

The transmission image, Fig. 3.29, is the first coefficient from **a**. The absorption image, Fig. 3.29, is calculated from

$$absorption = -\ln(\frac{sample}{reference}) \quad . \tag{3.15}$$

For a neutron interferometry absorption image, the low flux and radiation damaged detectors may require an `If` statement to prevent divide by zero problems. In the present

work, the statistics of the absorption image benefit from the acquisition of 16 images (the number of grating steps); the standard errors for the transmission coefficients are small, as listed in Fig. 3.9.

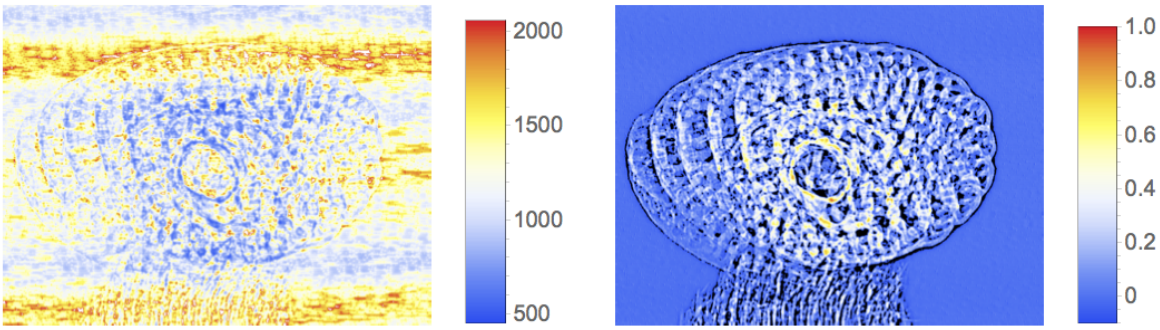


Figure 3.11: (left) The transmission image with a colorbar representing fitted detector counts. (right) The absorption image. The background appears smooth at this plot range, but be prepared for a surprise.

While Fig. 3.29 has a high contrast to noise ratio,

$$CNR = \frac{|S_A - S_B|}{\sigma_o} = \frac{|0.9 - 0.09|}{0.013} = 60, \tag{3.16}$$

it could be better. If we examine the background absorption more closely, Fig. 3.12, there is a noise feature we attribute to the phase and analyzer gratings. Both gratings have support structures that interrupt the ideal linear grating features. In principle, the grating features should have canceled in the calcaluation of the absorption, Eq. 3.15; the fact that cancellation did not occur suggests a slight motion of the interferometer during the time interval between the acquisition of the reference and absorption interferograms.

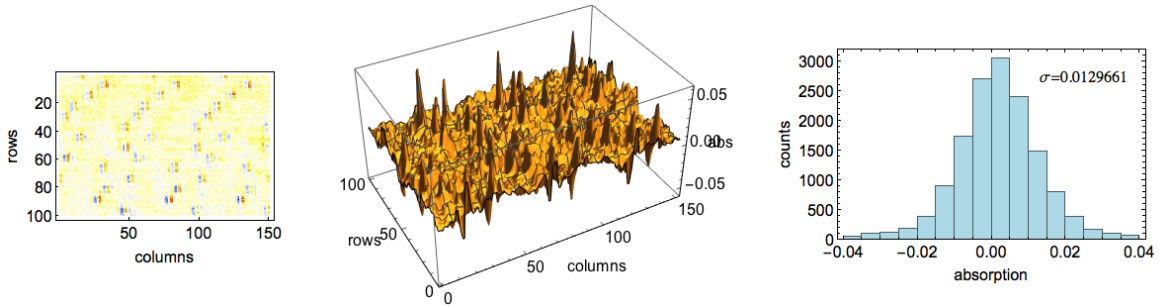


Figure 3.12: (left) The upper left corner of the absorption image, Fig. 3.29 is examined in detail. Structures with a pattern similar to the support structures in the gratings are visible in the color image and (middle) surface plot. (right) A histogram shows the standard deviation from this effect is affecting the contrast-to-noise of the absorption image.

A trace-back from the absorption to the transmission images, Fig. 3.13 confirms that most variations in the sample transmission are effectively normalized with the reference transmission, with the exception of near single-pixel features. The data were acquired with an effective pixel size of 0.645-μm. A brief exploration with row/column shifts of the sample transmission array indicates the peaks and valleys in the difference plot are strongly affected by ± 1 pixel shifts. However, a fractional pixel shift algorithm [3] is needed to prevent fuzzy air.

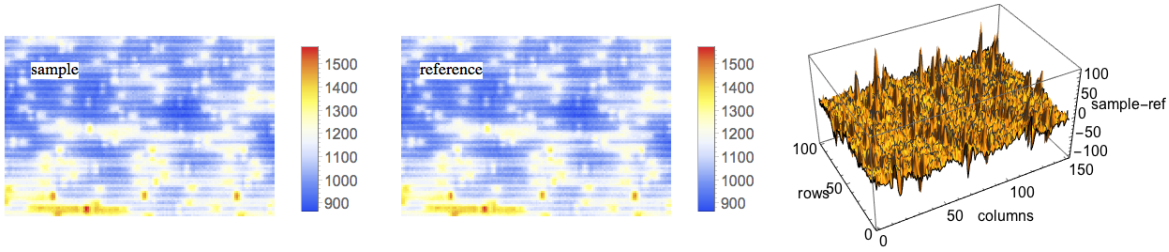


Figure 3.13: (left) The sample transmission data corresponding to the upper left corner of Fig. 3.29; (middle) and the corresponding section of the reference transmission data. (right) A difference plot shows the sample-reference transmission. Sub-pixel motion in the interferometer has increased the standard deviation of the backgroun, thus decreasing the image contrast-to-noise ratio. A fractional pixel shift algorithm applied to the sample transmission data is required to correct for the motion.

[3] `http://image-registration.readthedocs.io/en/latest/`

3.2.4 Amplitude Images

The amplitude of the sample interferogram,

$$amplitude = a_p = \sqrt{a_{2p}^2 + a_{3p}^2}, \tag{3.17}$$

is shown in Fig. 3.14. The amplitude image for a sample with low absorption typically shows little detail to the eye.

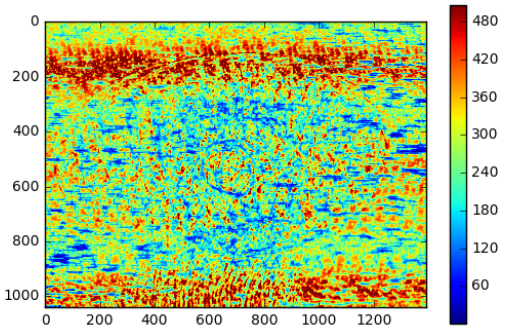


Figure 3.14: The amplitude image with colorbar in units of detector counts.

It is often more useful to plot the amplitude as normalized by the transmission, in percentage units, to give the percent visibility:

$$\%\text{visibility} = 100 \times \frac{amplitude}{transmission}, \tag{3.18}$$

Fig. 3.15 shows the percent visibility for the sample interferogram. Ideally, one hopes for visibility near 100% in air regions near the sample. Then, in regions where the sample absorbs X-rays or attenuates the neutron beam, we expect reduced visibility; as it happens, the expected reduction in percent visibility is scarcely detectable in this image where the maximum sample absorption is about 0.4 (see Fig. 3.29). We note that other effects which may diminish visibility include grating imperfections and small angle scattering. In fact, the hoped-for image in Fig. 3.15 is a nearly homogeneously gray image; the striations in the measured visibility in Fig. 3.15 are a sign of grating imperfections observable at an

effective pixel size of 0.645-μm. It is probable that a larger pixel size, or binning, would average over the grating imperfections.

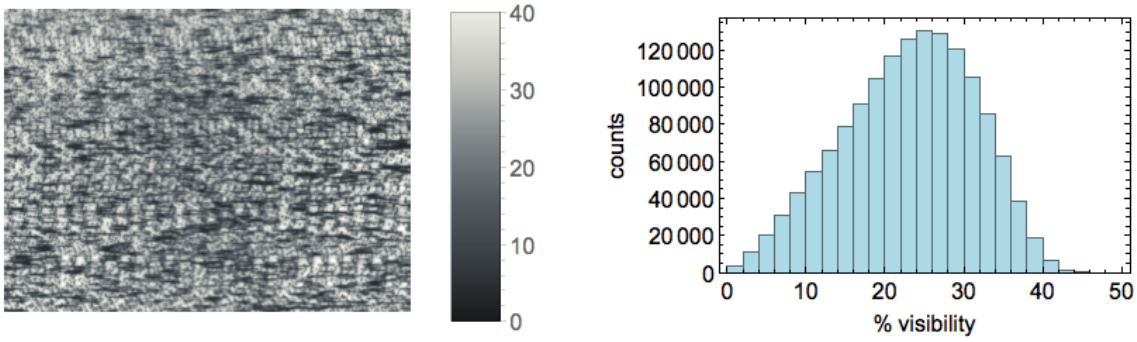


Figure 3.15: (left) The visibility image obtained with an analysis using a 0.48 μm grating step increment. The colorbar shows visibility in percentage units. (right) The histogram shows a most probable percent visibility near 25%. The dark striations in the figure are attributed to grating imperfections and show up in the histogram as counts at low visibility.

The interferometry/tomography experimentalist is often encounters unintended rotation and translation stage motions. At this writing, the last time one of us encountered a mis-performing stage motion was two weeks ago; at a new beamline at a national lab, a rotation stage performed 2.5$\times$ the intended rotation. We note that sub-micron grating motions are difficult to verify with measurement tools at the beamline. Thus, we are forced to use the data analysis software to verify grating motion. For comparison with Fig. 3.15, the visibility image is also shown as calculated with an intentional error in the grating step increment. The experiment was performed with 16 grating steps taken with an increment of 0.48 μm. If we analyze the interferogram with an incorrect grating increment, say 0.40 μm, the visibility should diminish, right? Yes, the sample visibility, see Fig. 3.16, is reduced, but the effect is subtle; we clearly need a better metric for overall interferogram fit and validation of the grating motion.

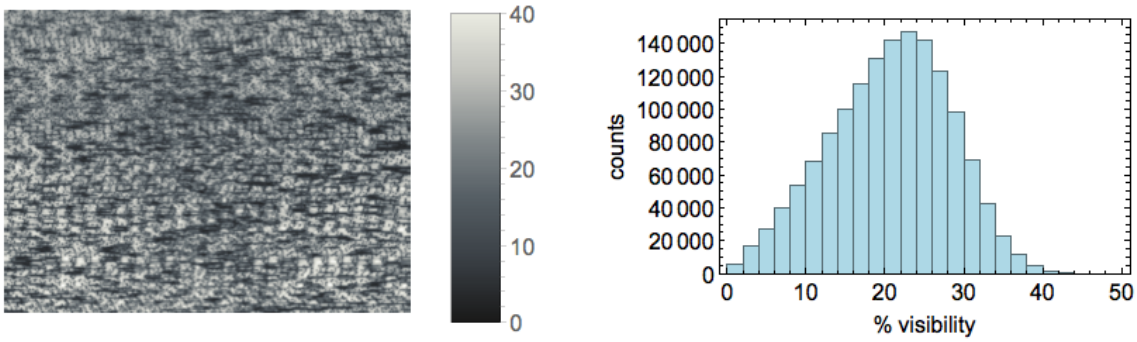


Figure 3.16: The surprise is the insensitivity of the visibility image to an incorrect analysis. (left) The visibility image obtained using an incorrect 0.40 μm grating step increment in the analysis for data that was acquired with 0.48 μm increment. The visibility image is little changed relative to the correct analysis, Fig. 3.15. (right) The histogram shows a noticeable reduction in visibility relative to the analysis with the correct grating period, Fig. 3.15.

3.2.5 Chi-Square Image

We can also assess the quality of the model fit to the interferogram with a plot of the reduced chi square, χ^2_ν,

$$\chi^2_\nu = \frac{1}{\nu}\sum_{g=1}^{M}\frac{(c_g - \hat{c}_g)^2}{\sigma_g^2} \tag{3.19}$$

$$= \frac{1}{M-3-1}\sum_{g=1}^{M}\frac{(c_g - \hat{c}_g)^2}{c_g} \tag{3.20}$$

where we have made the assumption that the standard deviation of the pixel measurements at each grating position is determined by shot noise, thus, $\sigma_g = \sqrt{c_g}$.

The expected value of χ^2_ν is near 1 for a good fit. As with visibility, we are using χ^2_ν to explore both quality of the experiment across the image and the quality of the grating increment. Figs. 3.17 shows a fit using the correct 0.48 μm grating step increment. The most probable value of χ^2_ν is near 1, indicate a good fit overall. However, striations are observed in the image as well as a long tail to high χ^2_ν values. Both the striations and the large χ^2_ν values are attributed to grating imperfections.

As with visibility, we explore the effect of performing the interferogram analysis with an

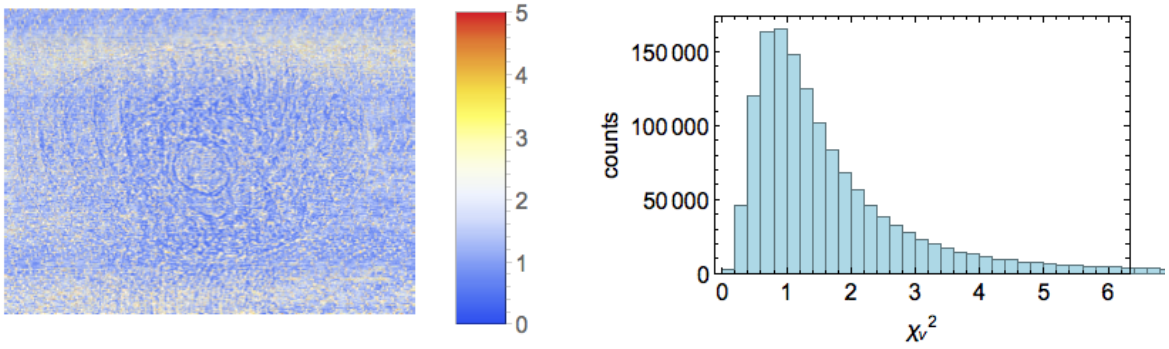


Figure 3.17: (left) Across most of the image, χ^2_ν is just over 1 indicating a good fit of a sinusoidal function to the interferogram. The grating increment is assumed to be 0.48 μm, as intended. (right) The striations in the image and the large χ^2_ν values in the histogram are attributed to grating imperfections.

incorrect grating increment. Fig. 3.18 shows the fit with the incorrect 0.40 μm value. There is a dramatic change in the image, much more than was apparent with the visibility images, Figs. 3.15 and 3.16 for correct and incorrect, respectively. Even better, the histogram in Fig. 3.18 shows a most probable χ^2_ν value well above 1 which is strong evidence of a poor fit. We recommend a reduced chi-square analysis as a check for the correct grating motion in a stepped grating experiment.

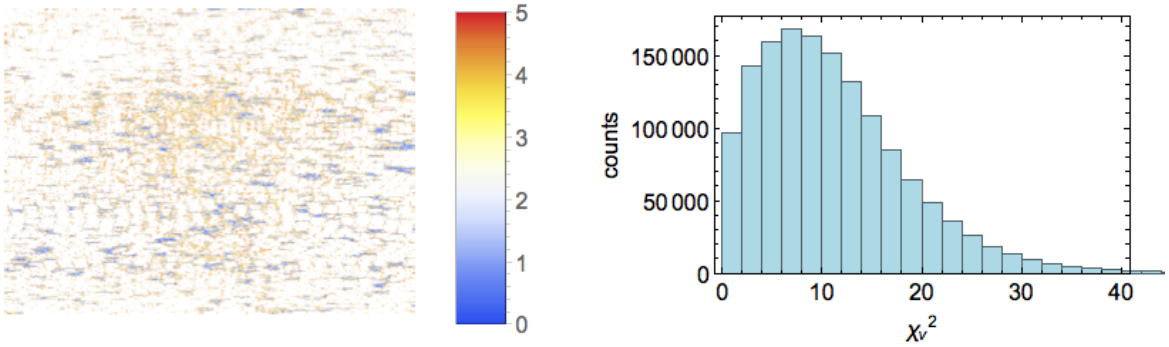


Figure 3.18: The reduced chi-square values for an analysis which used the incorrect value of the grating increment, 0.40 μm instead of the correct 0.48 μm. (left) Across most of the image, χ^2_ν is well over 5 indicating a poor fit. (right) The histogram shows the most probable χ^2_ν is near 8, indicating a poor fit.

If we analyze the upper left corner of the image with the χ^2_ν, Fig. 3.19, we see that this metric has the most sensitivity to the presumptive interferometry motion.

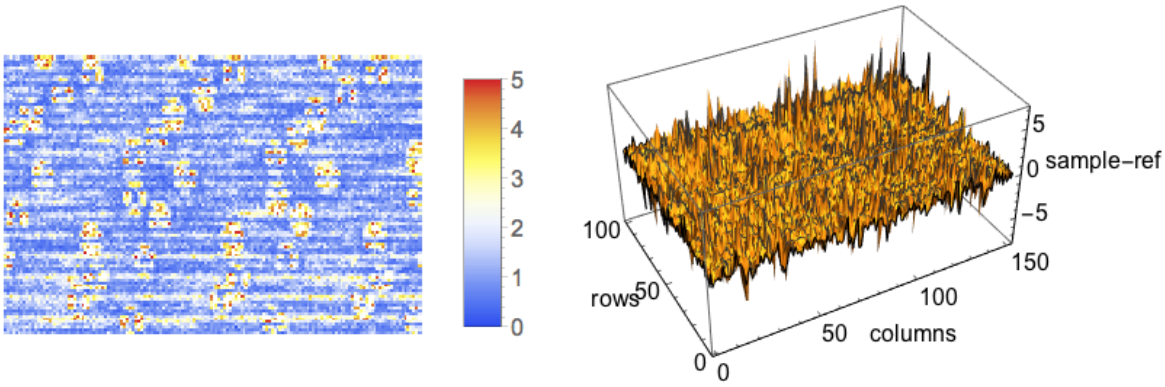


Figure 3.19: (left) A detail from the upper left corner of χ^2_ν and (right) a surface plot of the sample data. The χ^2_ν has more sensitivity than any other image modality or parameter to the effect of interferometry motion and grating support structure. It is this parameter that should be minimization objective for a sub-pixel motion of the sample data relative to the reference data.

3.2.6 Dark-Field Image

In optical microscopy, dark-field imaging is a method for observing low absorbing structures in the presence of high absorption, based on changes in the refractive index.[4] In the grating interferometry, the net result is nearly the same, hence the label of dark-field. It is also true that small angle X-ray (and neutron) scattering lead to image contrast, so one could call this the scattering image. Either way, the image is calculated as

$$dark-field = scattering = \frac{amplitude_{sample}/amplitude_{reference}}{transmission_{sample}/transmission_{reference}} \tag{3.21}$$

and should range from 0 to 1.

The foram dark-field image, Fig. 3.32, is a very poor quality image, one of the worst dark-field images ever obtained (and published) in grating-based X-ray interferometry. The striations in the background and across the sample are the same as in the visibility and χ^2_ν images. The grating imperfections are dominating the scattering signal.

There is strong interest in the dark-field image.

- brown adipose tissue [5]

[4]Molecular Expressions, Optical Microscopy Primer, Darkfield Illumination: `https://micro.magnet.fsu.edu/primer/techniques/darkfield.html`

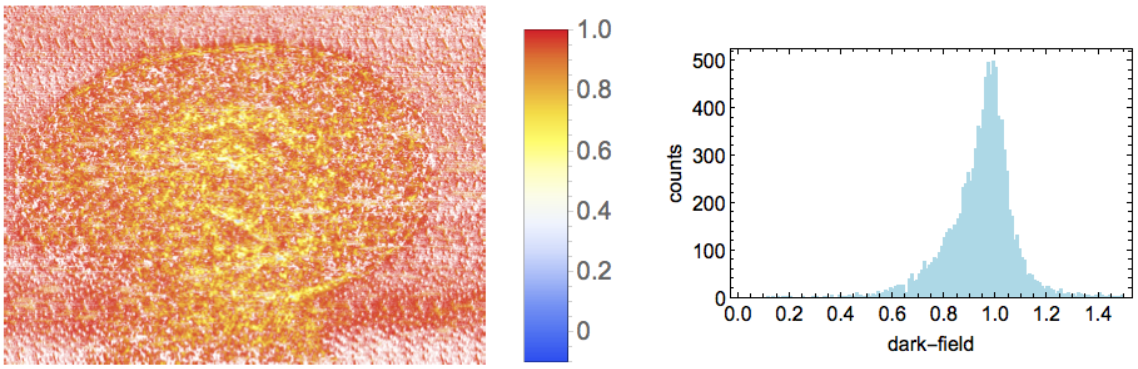


Figure 3.20: (left) The dark-field image and corresponding (right) histogram. Dark-field values above 1 are unexpected and are attributed to grating imperfections.

- lung disease [6]
- detection of sub-pixel defects in materials and devices [7]
- anisotropic materials [8]

The calibration studies are

- orientation dependence [9]
- quantification with known small particles [10]

The method has been extended by

- comparison to small angle scattering [11]
- theory [12]

3.2.7 Phase and Differential Phase Images

The foram differential phase image, Fig. 3.21, shows a wealth of structure within the sample, but this image is also compromised by the noise from the grating imperfections and interferometry motion. For example, the baseline DPC value in the air region should be 0 radians; deviation from zero is likely due to interferometry motion.

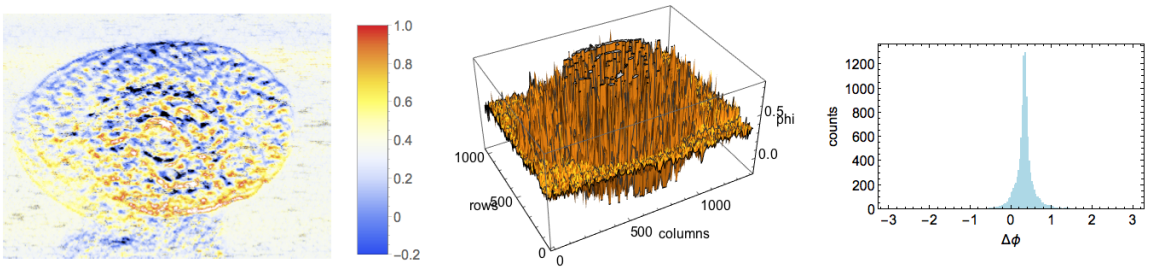


Figure 3.21: (left) The differential phase contrast image of the foraminifera. The colorbar is $\Delta\phi$ in radians. (middle) Surface plot of $\Delta\phi$. (right) Histogram. The min/max of the data extended to $\pm 2\pi$; the plot ranges have been reduced to show more detail.

The differential phase shift is simply

$$\Delta\phi = \phi^{sample} - \phi^{reference} \tag{3.22}$$

The ϕ^{sample} shows, Fig. 3.26, a sloping background which is of no significant consequence, so long as the interferometer is stable between the time of the reference and sample interferograms. One way to assess that stability is to inspect the $\Delta\phi$ in the air regions; it should be zero, which is nearly true as seen in Fig. 3.21.

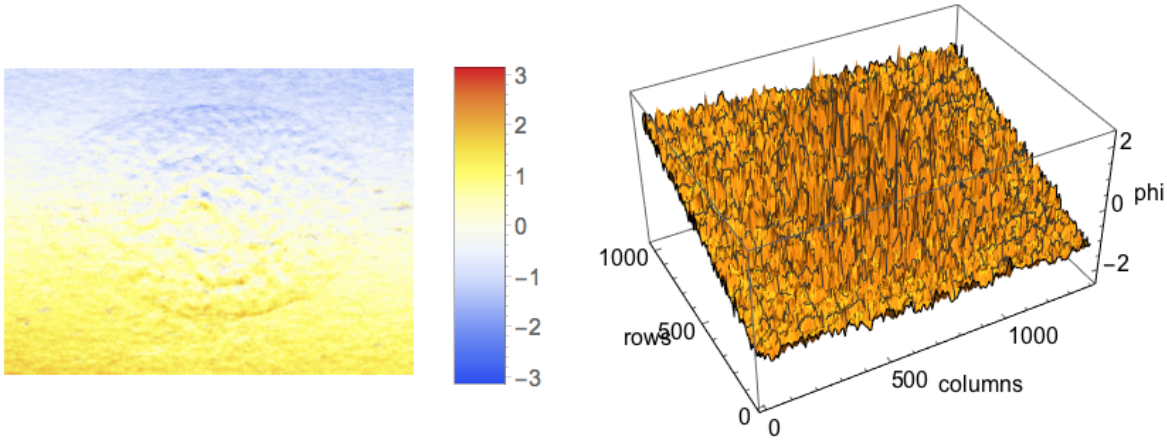


Figure 3.22: (left) An image of ϕ^{sample} and (right) a surface plot of the sample data. ϕ^{sample}. The full range of ϕ^{sample} values, lying in the range $[-\pi, \pi)$ are shown.

Zooming into the upper left corner of the foraminifera ϕ^{sample} image shows much structure from the gratings, Fig. 3.33.

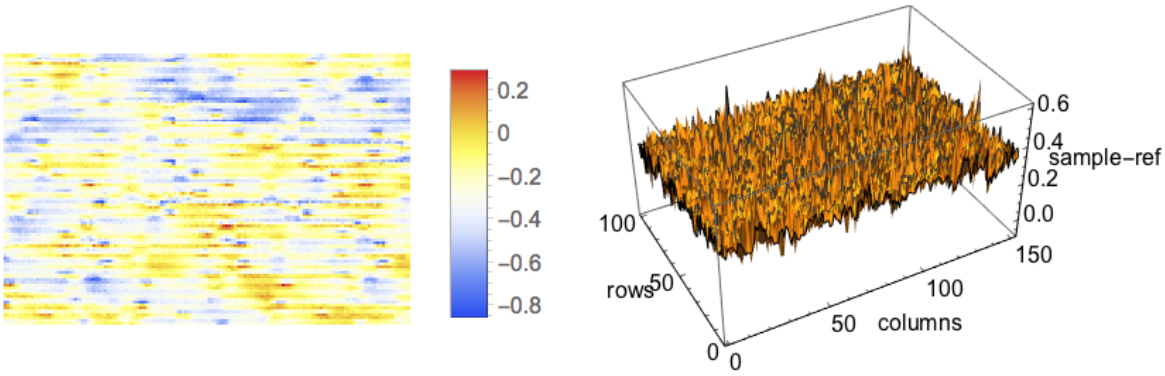


Figure 3.23: Foraminifera: (left) A detail from the upper left corner of ϕ^{sample} and (right) a surface plot of the sample data. The full range of ϕ^{sample} values, lying in the range $[-\pi, \pi)$ are shown. The small-scale structure in both images is attributed to the grating support structures. The origin of the large scale structure is unknown.

3.3 Stepped-Grating Interferometry in Python

The Mathematica code, Sec. 3.2, for the analysis of stepped-grating interferometry data has been converted to Python 3 [5] and enhanced. Our stepped-grating interferometry Python scripts have been incorporated into TomoPy package [6], an open source for data processing and image reconstruction in tomography.

3.3.1 Define Functions

Recall Eqs. 3.6, 3.7, and 3.8, which are repeated below.

$$\begin{aligned}
\hat{c}_{gp} &= a_{1p} + a_p \sin\left(\frac{2\pi}{p_{grat}} x_g + \phi_p\right) \\
&\equiv [1]\, a_{1p} + \left[\sin\left(\frac{2\pi}{p_{grat}} x_g\right)\right] a_p \cos(\phi_p) + \left[\cos\left(\frac{2\pi}{p_{grat}} x_g\right)\right] a_p \sin(\phi_p) \\
&\equiv \sum_{\mu=1}^{3} B_{g\mu} a_{\mu p} \\
&\quad a_{2p} \equiv a_p \cos(\phi_p), \qquad a_{3p} \equiv a_p \sin(\phi_p),
\end{aligned}$$

The second line shows how a basis set can be constructed with terms such as $[1]$, $\left[\sin\left(\frac{2\pi}{p_{grat}} x_g\right)\right]$, and $\left[\cos\left(\frac{2\pi}{p_{grat}} x_g\right)\right]$. The following Python code generates, numerically, these basis vectors as a function of x_g.

[5] https://github.com/jyuan4/Stepped_Gratting_Interferometry
[6] https://tomopy.readthedocs.io/en/latest/

```
def funcPrepareBvectorArbitrarySteps(gratingPeriodMicron, listGratingStepsMicron):
    numberGratingSteps = len(listGratingStepsMicron)
    b1 = np.ones((numberGratingSteps,1), dtype = np.int).flatten()
    b2 = np.sin(2 * math.pi * listGratingStepsMicron / gratingPeriodMicron)
    b3 = np.cos(2 * math.pi * listGratingStepsMicron / gratingPeriodMicron)
    return np.round(np.transpose(numpy.vstack([b1,b2,b3])),6)
bVector = funcPrepareBvectorArbitrarySteps(gratingPeriodMicron, listGratingStepsMicron)
```

and then the important **G** matrix is calculated with Eq. 3.11 and has dimensions of 3×M as the three basis vectors are defined and x_g undergoes M steps. In Python, we mainly use `numpy` module for vector calculation.

3.3.2 Calculate Coefficients for Reference Images

The near one-to-one translation of Mathematica to Python is seen in this function definition which includes a `for` loop for assembling a 2D interferogram, include the correction for beam off, i.e., dark field or detector noise image. We note that `TomoPy` is moving to broadly useful input module, `dxchange` for HDF5 reading; the `Image.open` is temporary.

```
def oneImageOneData(fname, dataDark):
    allData = np.zeros((rows, columns, numberGratingSteps))
    for k in range(numberGratingSteps):
        oneImage = Image.open(fname[k])
        oneData = np.array(oneImage) - dataDark
        allData[:,:,k] = oneData
    return allData
allReferenceData = oneImageOneData(filenamesReference, dataDark)
```

The vectorization relies on array reshaping commands, both their convenience and speed. An interactive development environment (IDE) such as Spyder[7] is ideal for this

[7]Spyder: `https://pythonhosted.org/spyder/`

code development as one must continuously assess array dimensions.

```
cVector = np.transpose(np.reshape(allReferenceData, (rows*columns,
numberGratingSteps)))
aVector = np.dot(gMatrix, cVector)
aMatrix = np.reshape(np.transpose(aVector), (rows, columns, 3))
refVisibility = np.sqrt(aMatrix[:,:,1]**2 + aMatrix[:,:,2]**2)
refPhi = np.arctan2(aMatrix[:,:,2], aMatrix[:,:,1])
refTransmission = aMatrix[:,:,0]
refVisibilityPercent = 100 * refVisibility / refTransmission
```

3.3.3 Calculate ChiSquare of the Sinusoidal Fit

The Mathematica version of this loop is annoying slow, but Python seems fast due to the compiler in Python. Thus, Python script for chi-square sinusoidal fit will be considered for model examination.

```
countsCalculated = np.dot(bVector, np.transpose(np.reshape(aMatrix,
   (rows * columns, 3))))
def chiSquare(cVector, countsCalculated):
    data = np.zeros(rows * columns)
    print (data.shape)
    for p in range(cVector.shape[1]):
        cg = cVector[:, p]
        cgHat = countsCalculated[:, p]
        data[p] = np.sum((cg - cgHat)**2 / cg) / (numberGratingSteps - 3 - 1)
    return data
start = time.time()
chiSquareRef = chiSquare(cVector, countsCalculated)
end = time.time()
```

```
chiSquareRef = np.reshape(chiSquareRef, (rows, columns))
```

The histogram is valuable, but can be slow for large images. Excellent results are still obtained by a random sampling of the image, and then plotting a histogram.

3.3.4 Calculate Coefficients for Sample Images

The parameters transmission, amplitude, and phi are defined in Eq. 3.8. If additional terms are needed to model anharmonic functions, the variable `aVector` will increase from 3×N to 4×N or 5×N or larger. The dot product operation will still be fast.

```
allSampleData = oneImageOneData(filenamesSample, dataDark)
cVector = np.transpose(np.reshape(allSampleData, (rows*columns, numberGratingSteps)))
aVector = np.dot(gMatrix, cVector)
aMatrix = np.reshape(np.transpose(aVector), (rows, columns, 3))
sampleVisibility = np.sqrt(aMatrix[:,:,1]**2 + aMatrix[:,:,2]**2)
samplePhi = np.arctan2(aMatrix[:,:,2], aMatrix[:,:,1])
sampleTransmission = aMatrix[:,:,0]
sampleVisibilityPercent = 100 * sampleVisibility / sampleTransmission
smallSamplePhi = scipy.ndimage.interpolation.zoom(scipy.signal.medfilt(samplePhi, 1),
  1/math.ceil(columns/200))
```

Here we use `scipy` module to calculate median filter. Especially, `scipy.signal.medfilt` performs a median filter on an N-dimensional array. The variable `smallSamplePhi` is used for plotting speed of a 3D surface plot. Fig. 3.24 shows two components of the sinusoidal fit, the transmission and amplitude. Next, these two parameters will be combined into visibility, and then percent visibility, Fig. 3.25. Percent visibility across the image is a valuable assessment of quality of an interferometry experiment.

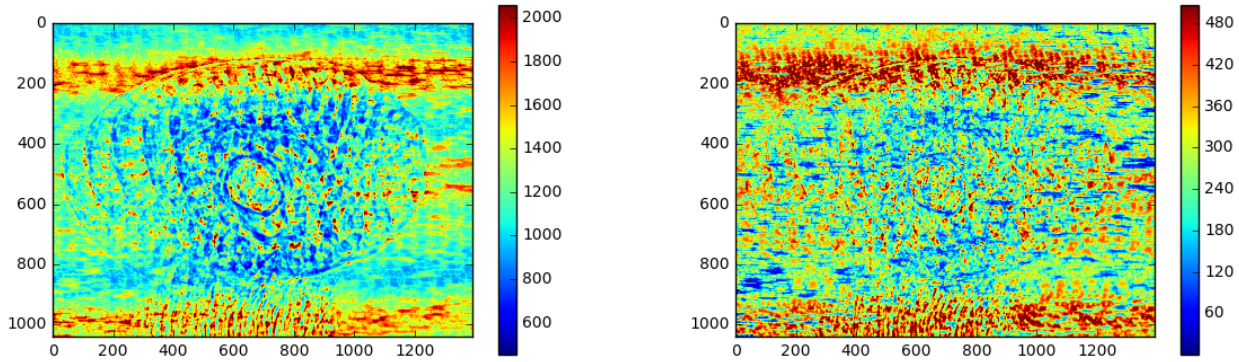


Figure 3.24: In Python, (left) The transmission image with a colorbar representing fitted detector counts. (right) The amplitude image with colorbar in units of detector counts.

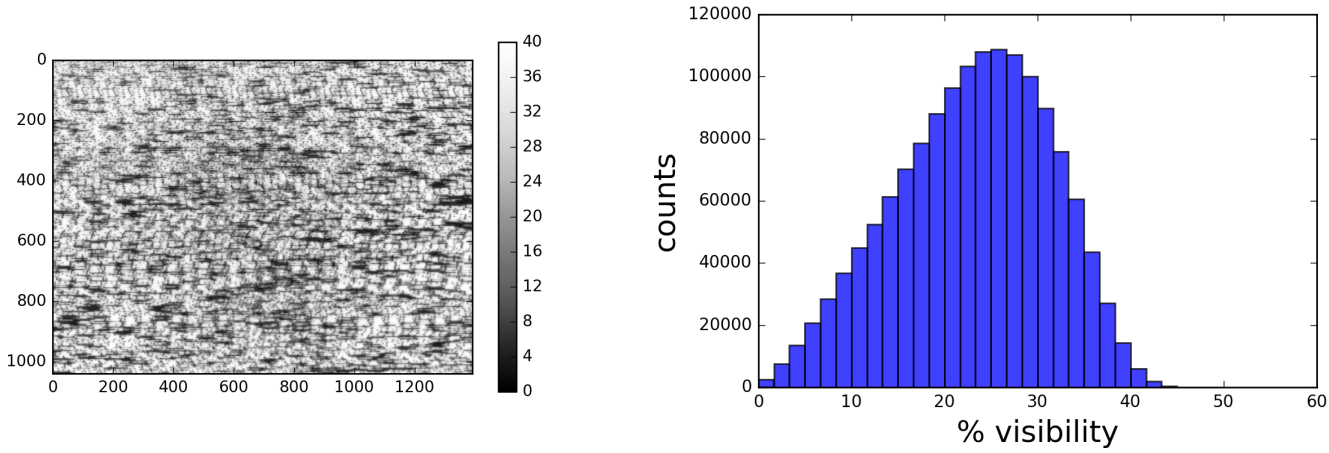


Figure 3.25: In Python, (left) The visibility image obtained with an analysis using a 0.48 μm grating step increment. The colorbar shows visibility in percentage units. (right) The histogram shows a most probable percent visibility near 25%. The dark striations in the figure are attributed to grating imperfections and show up in the histogram as counts at low visibility.

3.3.5 Calculate ChiSquare of the Sinusoidal Fit

When calculating `chiSquareSample`, we continue using `numpy` for vector performances.

```
countsCalculated = np.dot(bVector, np.transpose(np.reshape(aMatrix,

(rows * columns, 3))))

start = time.time()

chiSquareSample = chiSquare(cVector, countsCalculated)

end = time.time()
```

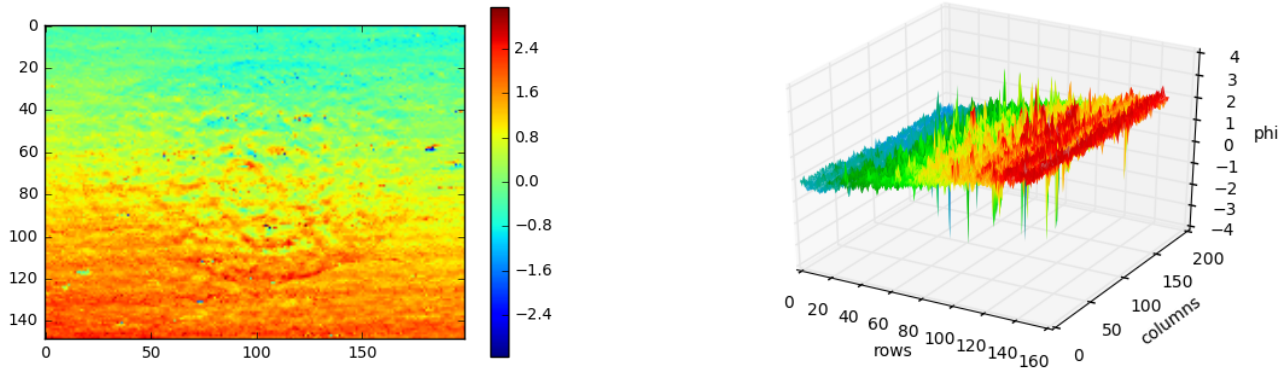


Figure 3.26: In Python, (left) An image of ϕ^{sample} and (right) a surface plot of the sample data. ϕ^{sample}. The full range of ϕ^{sample} values, lying in the range $[-\pi, \pi)$ are shown.

```
chiSquareSample =np.reshape(chiSquareSample, (rows, columns))

print (new_chiSquareSample.shape)
```

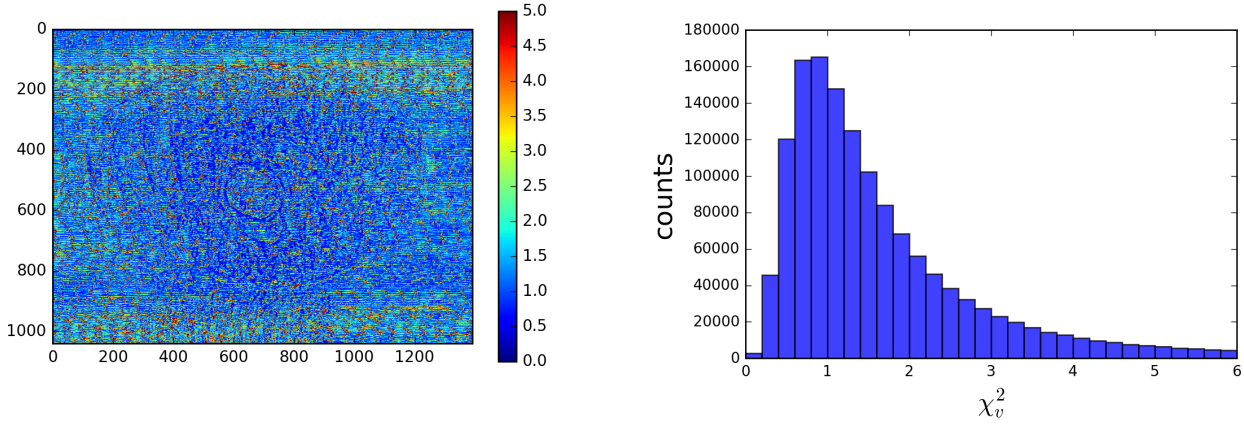


Figure 3.27: In Python, (left) Across most of the image, χ^2_ν is just over 1 indicating a good fit of a sinusoidal function to the interferogram. The grating increment is assumed to be 0.48 μm, as intended. (right) The striations in the image and the large χ^2_ν values in the histogram are attributed to grating imperfections.

3.3.6 Calculate Best Fit with NLM and Linear Algebra

There are no plenty of options in Python non-linear regression model, so we created the non-linear fitting model function by ourselves. With least-square error method, we defined sum of error, bounds of variables, initial values, and method `L-BFGS-B`. The optimal result is to minimize these conditions.

```
def NonLinearModelFit(x, *p):
    a, b, c = p
    return a + b * np.sin(x * (2 * math.pi / gratingPeriodMicron) + c)
def fitNLM(p0, x, y_noise, p_init):
    err = lambda p: np.mean((NonLinearModelFit(x, *p)-y_noise)**2)
    p_opt = minimize(
                err,
                p_init,
                bounds = [(None, None), (0, None), (-math.pi, math.pi)],
                method="L-BFGS-B"
                ).x
    return p_opt
```

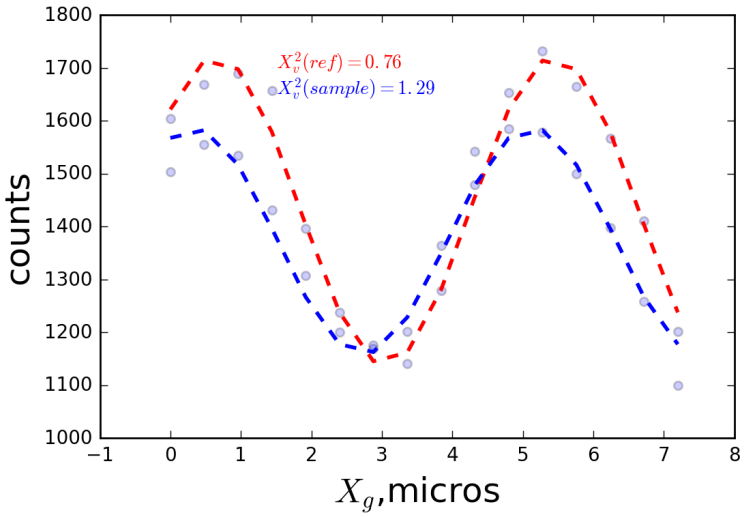


Figure 3.28: In Python, a non-linear least squares fit to Eq. 3.1. The results are identical to the linear algebra procedure and code used to prepare Fig. 3.8. The confidence intervals are at the 95% level. The top table gives the fit parameters for the reference interferogram (red) and the bottom table is for the sample (blue).

Here, we got χ^2 for reference is equal to 0.76 and 1.29 for sample data, both of which are pretty low, a good fit for the non-linear regression model.

3.3.7 Calculate Absorption Image

We calculated `absorption` image through Beer's Law.

```
absorption = -np.log(sampleTransmission / refTransmission)
```

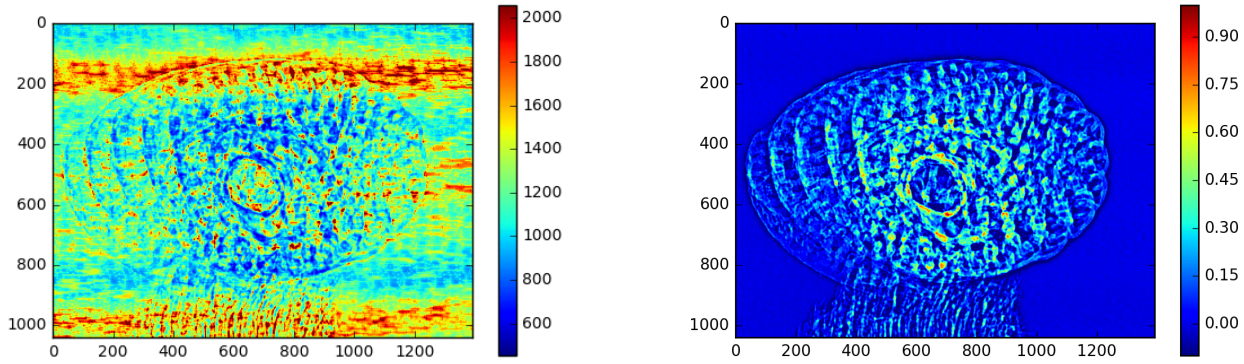


Figure 3.29: In Python, (left) The transmission image with a colorbar representing fitted detector counts. (right) The absorption image. The background appears smooth at this plot range, but be prepared for a surprise.

3.3.8 Calculate Differential Phase Contrast Image

When calculating differential phase contrast images, we downsized the variable `sampleDPC` and applied median filter with `scipy` module.

```
differentialPhase = samplePhi - refPhi
smallDPC =  scipy.ndimage.interpolation.zoom(scipy.signal.medfilt(differentialPhase, 1),
1/math.ceil(columns/200))
```

3.3.9 Calculate Dark-field Image

Dark field image was calculated with sample/reference visibility array and sample/reference transmission array.

```
darkfield = (sampleVisibility / refVisibility) / (sampleTransmission / refTransmission)
```

In the next step, we cropped sample transmission array to get more specific information as shown 2D and 3D images below. Moreover, cropped absorption, cropped phi and cropped visibility were calculated in the following figures.

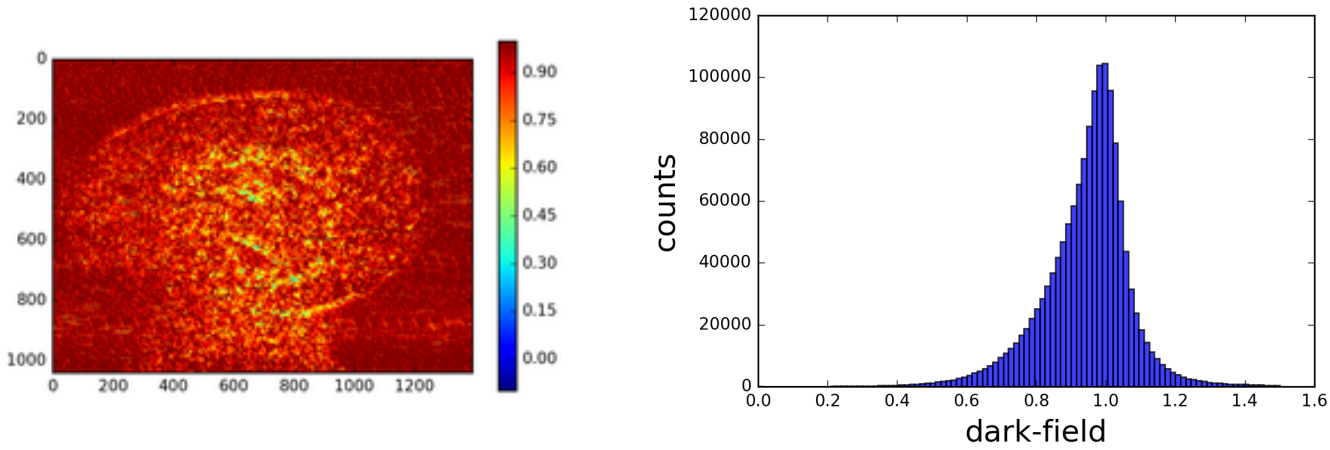


Figure 3.30: In Python, (left) The dark-field image and corresponding (right) histogram. Dark-field values above 1 are unexpected and are attributed to grating imperfections.

```
cropSampleTransmission  = sampleTransmission[cropLimitRows[0]:cropLimitRows[1],
cropLimitColumns[0]:cropLimitColumns[1]]
```

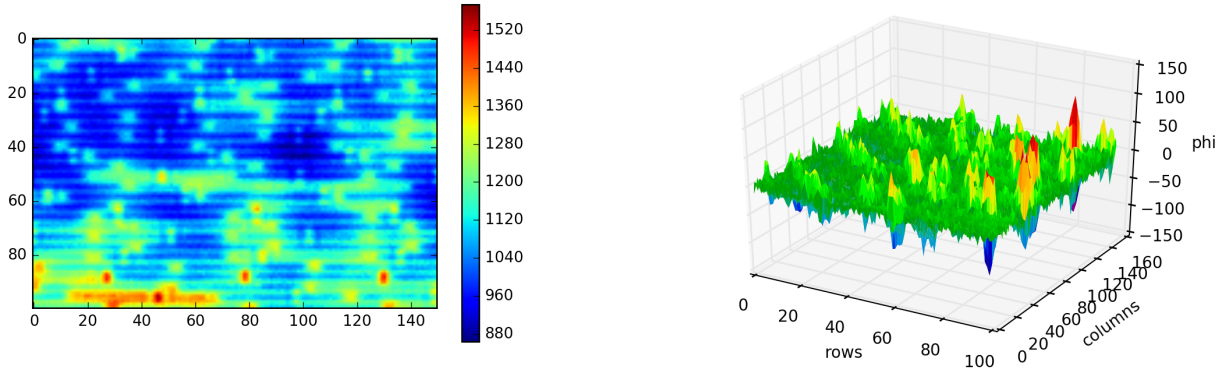


Figure 3.31: In Python, the micro-structure Foraminifera: (left) $\phi^{reference}$ and (right) a surface plot of the sample data. The full range of ϕ^{sample} values, lying in the range $[-\pi, \pi)$ are shown. The small-scale structure in both images is attributed to the grating support structures. The origin of the large scale structure is unknown.

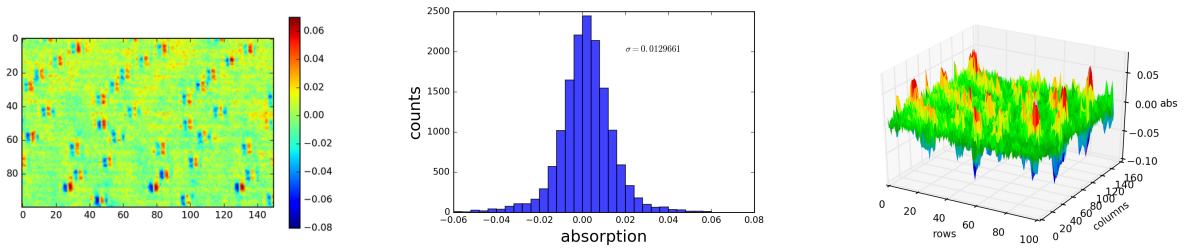


Figure 3.32: In Python, (left) The dark-field image, (middle) histogram and corresponding (right) histogram. Dark-field values above 1 are unexpected and are attributed to grating imperfections.

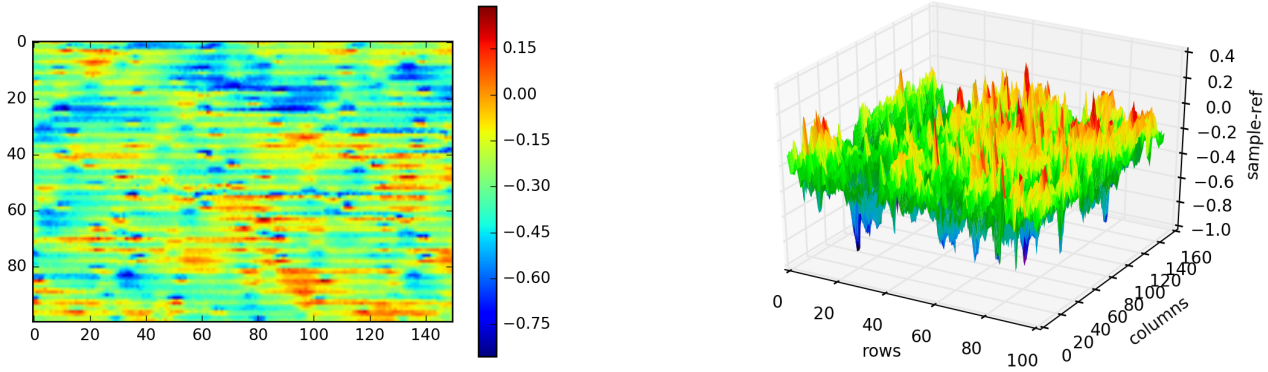


Figure 3.33: Foraminifera: (left) A detail from the upper left corner of ϕ^{sample} and (right) a surface plot of the sample data. The full range of ϕ^{sample} values, lying in the range $[-\pi, \pi)$ are shown. The small-scale structure in both images is attributed to the grating support structures. The origin of the large scale structure is unknown.

3.4 Single-shot Interferometry

This section describe one strategy for processing single-shot interferometry data acquired with a checkerboard phase grating. The Mathematica code was developed at LSU based largely on the publication by Dr. Han Wen and co-workers at NIH.[13, 14, 15, 16, 17]

The code has been used to process data acquired at APS to generate movies and tomography of flame retardants in polymer blends.[18] Also acquired at APS by Dr. Shashi Marathe are images of 100 to 225 μm polystyrene spheres affixed to a Kapton film. This data will be used illustrate the processing of single-shot, checkerboard phase grating interferomtry data.

The processing sequence is:

1. Secs. 3.4.1 to 3.4.6: Fourier transform the reference image and begin the extremely tedious process of locating harmonics. In the Fourier space for the reference and sample image, six harmonics will be located to single pixel accuracy: H_{00}^{ref}, H_{10}^{ref}, H_{01}^{ref}, H_{00}^{sample}, H_{10}^{sample}, and H_{02}^{sample}. A {row,column} subscript notation is used to identify the central, vertical, and horizontal harmonics, as described in Fig. 3.34.

2. Sec. 3.4.7: The six harmonics are processed into six images. The processing is fast and involves application of a Hanning filter to reduce Fourier transform truncation errors, zero filling to yields images of a consistent dimension, and an inverse Fourier transform.

3. Sec. 3.4.8 to 3.4.10 describe the simple process of generating absorption, dark-field, and differential phase contrast (DPC) images.

4. Sec. 3.4.11 describes phase unwrapping as guided by the absorption image. This is unique as most phase unwrapping algorithms use only the phase image. In principle, an absorption guided process should be robost, but the results are only moderately successful.

5. Sec. 3.4.12 The differential phase contrast 2D images are visually interesting, but the quantitative processing is more easily performed on the integrated result, a phase image. The Frank-Chellapa algorithm is applied with modest success.

3.4.1 Raw Data and FFT

An FFT of any image will give a central peak representing the image intensity. When there is an underlying pattern, such as the grid pattern in the raw image, then harmonics are also visible. We are interested in the central harmonic, H_{00}, the harmonic above the central, H_{10}, and the harmonic to the right of the central, H_{01}, where we use the {row,column} subscript notation.

If the checkerboard is not aligned with the pixelated detector, the harmonics will appear rotated. Thus, an early task is location of the harmonics, computing the rotation error, and rotating all data before the Fourier transform.

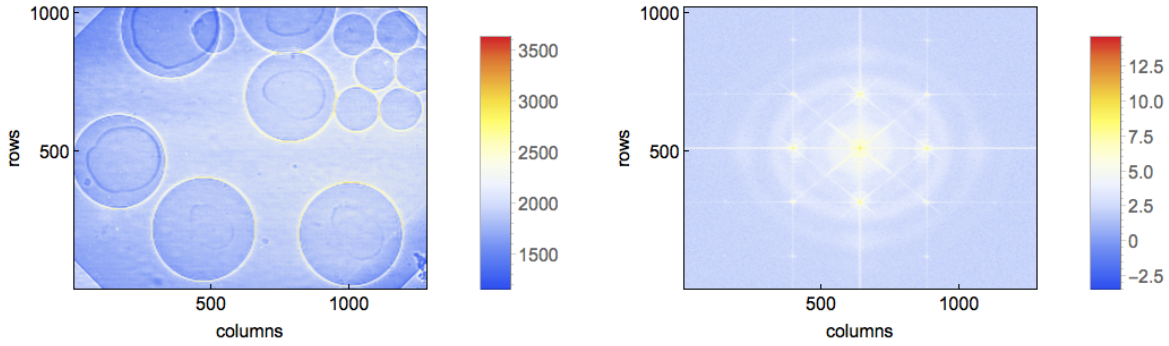


Figure 3.34: (left) Raw data of polystyrene spheres on a Kapton film. (right) The $\ln|\mathcal{F}(ref)|$ of the reference image. We are interested in the central harmonic, H_{00}, the harmonic above the central, H_{10}, and the harmonic to the right of the central, H_{01}, where we use the {row,column} subscript notation. Later, superscripts of `ref` and `sample` will indicate the respective data sets.

The Mathematica code uses the `FourierParameters -> {0, 1}` to ensure that the transform is performed with the discrete version of $\frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)\exp^{i\omega t}\,dt$. The zero-lag (zero frequency) element in the list of a 1D or a corner element in a 2D image. Thus, some list rearrangement is needed. In Matlab, this is the `fftshift` command. In Mathematica,

```
dataFFT=Fourier[data,FourierParameters->{0,1}];
dataFFT=RotateLeft[dataFFT,Round[rows/2]];
dataFFT=Transpose[RotateLeft[Transpose[dataFFT],Round[columns/2]]];
```

3.4.2 Estimate Period

Initially, the distances between the harmonics for a checkerboard phase grating are estimated based on the grating period, the effective pixel size, and the data dimensions, {rows, columns}:

$$period_{estimated} = \frac{\sqrt{2}(\text{pixel size})(\text{rows, columns})}{\text{grating period}} \tag{3.23}$$

3.4.3 Initial Coordinates

The code for the initial coordinate determination of the harmonics is listed below. The basic idea is this:

1. Estimate coordinates based on Eq. 3.23

2. Develop crop regions around the harmonics

3. Extract image data for each harmonic. Note, we are using $\ln|\mathcal{F}(ref)|$ of the reference image. The log, absolute value is to make the image real, positive. The reference image has less peak structure than the sample image. These factors make the peak positions of the harmonics easier to locate.

4. Within each region, find the maximum value, then find the {row,column} coordinates of that pixel.

5. Verify {row,column} coordinates with line probe plots such as in Figs.3.37, 3.38, and 3.39. Fudge the coordinates with addition of $\{\pm 1, \pm 1\}$ as needed. No fudge factors were needed with this code, but details of the image cropping function may lead to small factors.

```
coord00 = Round[{rows,columns}/2];
coord10 = (coord00 + {-periodVertical,0})
coord01 = coord00 + {0,periodHorizontal};
rangeRows = {1,periodVertical}-Ceiling[periodVertical/2];
rangeColumns= {1,periodHorizontal}-Ceiling[periodHorizontal/2];
cropHarmonic00 = {coord00[[1]]+rangeRows, coord00[[2]]+rangeColumns} ;
cropHarmonic10 = {coord10[[1]]+rangeRows, coord10[[2]]+rangeColumns} ;
cropHarmonic01 = {coord01[[1]]+rangeRows, coord01[[2]]+rangeColumns} ;
region00= Take[lnAbsDataFFTref,cropHarmonic00[[1]],cropHarmonic00[[2]] ];
region10= Take[lnAbsDataFFTref,cropHarmonic10[[1]],cropHarmonic10[[2]] ];
region01= Take[lnAbsDataFFTref,cropHarmonic01[[1]],cropHarmonic01[[2]] ];
```

```
coord00=Flatten[Position[region00, p_?(#==Max[region00]&)]]
                + Round[{rows,columns}/2]-Round[{periodVertical,periodHorizontal}/2]+{0,0}
coord10=Flatten[Position[region10, p_?(#==Max[region10]&)]]
                + Round[{rows,columns}/2]-Round[{3*periodVertical,periodHorizontal}/2]+{0,0}
coord01=Flatten[Position[region01, p_?(#==Max[region01]&)]]
                + Round[{rows,columns}/2]-Round[{periodVertical,-periodHorizontal}/2]+{0,0}
```

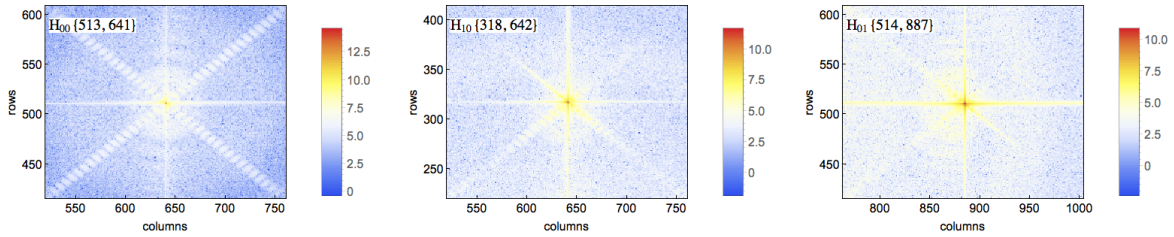


Figure 3.35: The extracted harmonics based on the estimated periods and taken from the $\ln|\mathcal{F}(ref)|$ of the reference image. (left) H_{00} (middle) H_{10} (right) H_{01}.

3.4.4 Period, Rotation Error, and Data Rotation

```
vectorVertical = coord10-coord00
vectorHorizontal=coord01-coord00
periodVertical=Round[N[Norm[vectorVertical]]]
periodHorizontal=Round[N[Norm[vectorHorizontal]]]
rotationErrorRadians =Mean[{N[VectorAngle[vectorVertical,{0,1}]]-\[Pi]/2,
                                    N[VectorAngle[vectorHorizontal,{0,1}]]}]
Print["rotation error = "<>ToString[rotationErrorRadians*180/\[Pi]] ]
```

The rotation error in Fig. 3.34 is small, only -0.03°. The `ImageRotate` has a few special options.

1. The image center is the center of rotation.
2. The size of the rotated image is {rows,columns}.
3. The pixel values are calculated with linear interpolation.

```
dataReference=dataRef-dataDark;
dataReference=ImageData[ImageRotate[Image[dataReference,"Real"],
        -rotationErrorRadians,{columns,rows},Resampling->"Linear"],"Real"];
dataSample=dataRaw-dataDark;
dataSample=ImageData[ImageRotate[Image[dataSample,"Real"],
        -rotationErrorRadians,{columns,rows},Resampling->"Linear"],"Real"];
```

3.4.5 Final Coordinates

Now that the data are rotated and the periods are known, please repeat Sec. 3.4.3 to get better coordinates for the harmonics.

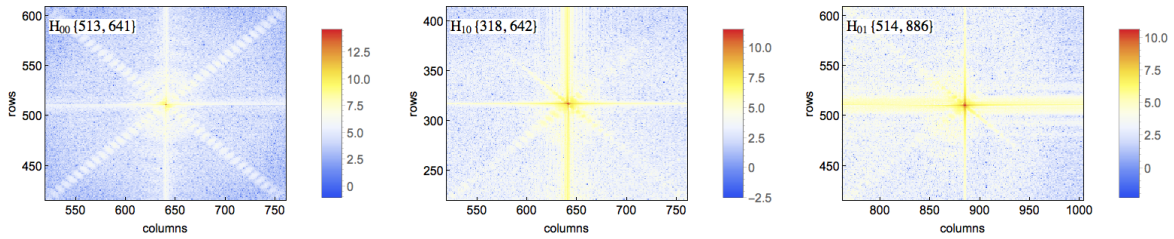


Figure 3.36: The extracted harmonics based on the estimated periods and taken from the $\ln|\mathcal{F}(ref)|$ of the reference image. (left) H_{00} (middle) H_{10} (right) H_{01}.

3.4.6 Line Probes

These line probes will verify harmonic coordinates obtained in Sec. 3.4.5 to 1 pixel accuracy. The peaks are labeled with the harmonic of interest in that probe. The line probes are also our first indication of the data quality. The similar amplitudes of the H_{00}^{ref} and H_{00}^{sample} traces indicates low sample absorption. The high frequency structure in H_{00}^{sample} indicates sharp edges in the sample. The near baseline separation of H_{00}^{sample} from both H_{10}^{sample} and H_{01}^{sample} indicates good resolution in the DPC and dark-field images.

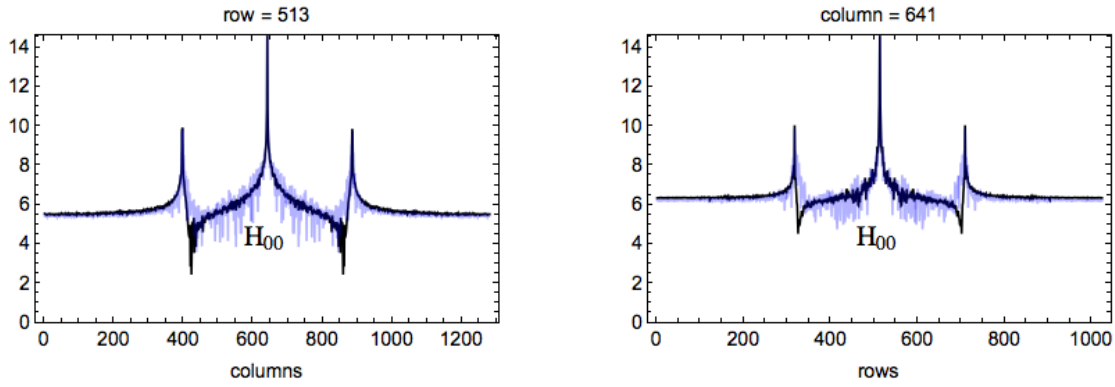


Figure 3.37: H_{00}: The black trace shows H_{00}^{ref} and the blue trace shows H_{00}^{sample}. The H_{00} {row,column} coordinates are given in the plot titles. The high frequency structure in the blue trace shows the sample structure. The amplitude of the blue trace is nearly as high as the black, indicating the sample has low absorption.

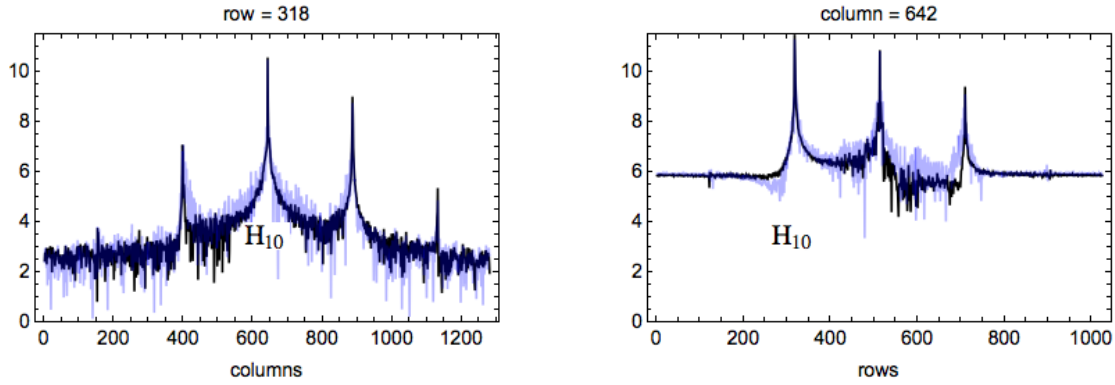


Figure 3.38: H_{10}: This is the harmonic above the central harmonic in Fig. 3.34. The {row,column} coordinates are correct when the labeled harmonic amplitudes are equal in the two line probes. We note the column coordinate for H_{10} may differ by ± 1 from H_{00}.

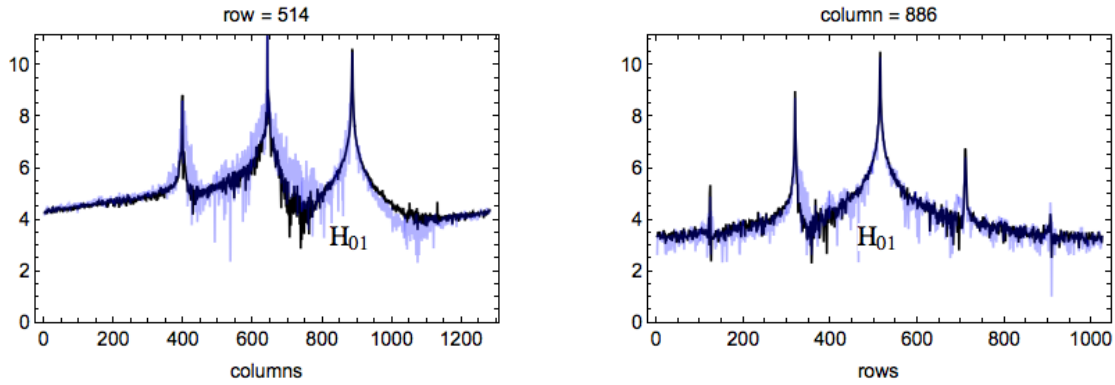


Figure 3.39: H_{10}: This is the harmonic to the right of the central harmonic in Fig. 3.34. The black trace shows H_{01}^{ref} and the blue trace shows H_{01}^{sample}. We note the row coordinate for H_{01} may differ by ± 1 from H_{00}.

3.4.7 Apply Hanning Filter, Zero Filling, and Inverse FFT

In this section, a harmonic in Fourier-transform space is converted to an image in real-space through a 2D inverse FFT in three steps. First, to reduce truncation wiggles in the iFFT, a Hanning weighting function will be applied to harmonic. Second, to yield images of the original {row,column} size, the harmonic will be zero filled with the harmonic peak positions assigned to the same coordinate as for the peak in H_{00}^{ref}. Third, the inverse Fourier transform will be applied with the same parameters as for the forward transform. This will yield six images; ratios of will yield five images: the absorption, differential phase contrast (vertical and horizontal), and dark-field images (vertical and horizontal).

- **Hanning Filter**

The 2D Hanning function is generated and applied without looping by taking advantage of vectorized element-by-element multiplication. As a Mathematica function, the code is:

```
funcHanningFilter[data_]:=Module[{rows,columns,hanningRows,hanningColumns,filteredData},
{rows,columns}=Dimensions[data];
filteredData=ConstantArray[0,{rows,columns}];
hanningRows = 0.5(1-Cos[ 2  Pi N[ Range[0,rows-1]/(rows-1)]]);
hanningColumns= 0.5(1-Cos[ 2 Pi N[ Range[0,columns-1]/(columns-1)]]);
filteredData=hanningRows *data;
filteredData=Transpose[hanningColumns*Transpose[filteredData]]  ]
```

- **Zero Filling**

Zero filling without looping is accomplished by generating lists of row and columns indices centered about the H_{00}^{ref} peak, defining an array of zeros with dimensions {rows,columns}, and using a list-based assignment of a small array into a larger array. The result should be verified by listing array values near the H_{00}^{ref} peak; small tweaks of ± 2 may be needed.

We note that the MRI (magnetic resonance imaging) literature has tutorials on the topics of filtering, zero filling, and Fourier transform.

- **Hanning Filter, Zero Filling, and iFFT**

The Mathematica code for filtering, zero filling, and the inverse Fourier transform as applied to H_{00}^{sample} to give image I_{00}^{sample} is listed here. Recall that `dataFFTsample` was first shown following Fig. 3.34 and `cropHarmonic` was introduced in Sec. 3.4.3.

```
region00=Take[dataFFTsample,cropHarmonic00[[1]],cropHarmonic00[[2]] ];
listRows=-1+coord00[[1]]-Ceiling[periodVertical/2]+Range[periodVertical];
listColumns=-1+coord00[[2]]-Ceiling[periodHorizontal/2]+Range[periodHorizontal];
shiftedPeak00=ConstantArray[0,{rows,columns}];
shiftedPeak00[[listRows,listColumns]]=funcHanningFilter[region00];
result00Sample=InverseFourier[shiftedPeak00,FourierParameters->{0,1}];
```

The above code is repeated a total of six times to generate I_{00}^{ref}, I_{10}^{ref}, I_{01}^{ref}, I_{00}^{sample}, I_{10}^{sample}, and I_{01}^{sample}. Please recall that the subscript notation is {row,column}. Soon, images with information about sample properties along the horizontal image axis will be generated. These images will require images with a non-zero column index such as I_{01}^{ref} and I_{01}^{sample}.

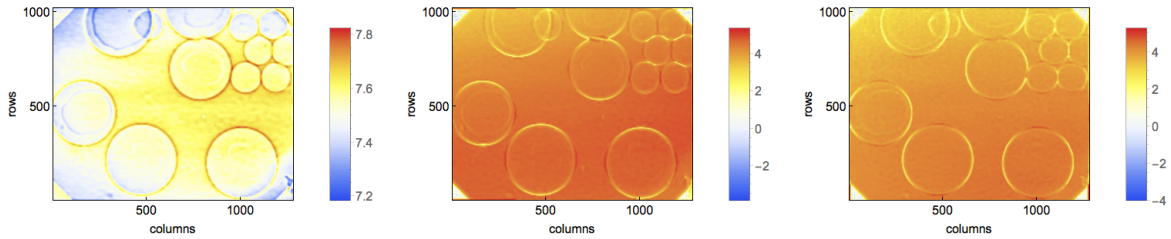


Figure 3.40: The $\ln|I|$ representations of (left) I_{00}^{sample} (middle) I_{10}^{sample} (right) I_{01}^{sample}

3.4.8 Absorption

The absorption is calculated from the complex images as

$$absorption = -\log\left[\left|\frac{I_{00}^{sample}}{I_{00}^{ref}}\right|\right] \tag{3.24}$$

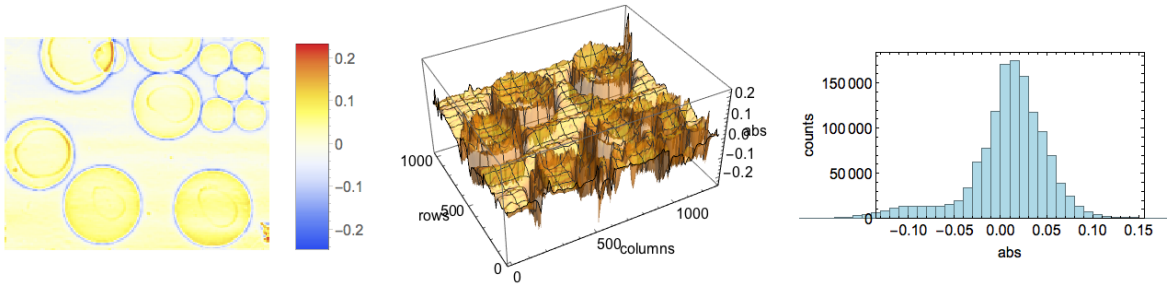


Figure 3.41: Absorption image: The sample is 100 and 225 μm diameter polystyrene spheres affixed to a Kapton film. The film adds a slight absorption offset. The fluid used to hold the spheres to the film is visible on some spheres.

3.4.9 Dark-Field or Scattering

The dark-field (or scattering) image is calculated as absorption normalized vertical and horizontal harmonics:

$$dark-field(vertical) = -\log\left[\left|\frac{I_{10}^{sample}/I_{10}^{ref}}{I_{00}^{sample}/I_{00}^{ref}}\right|\right] \tag{3.25}$$

$$dark-field(horizontal) = -\log\left[\left|\frac{I_{01}^{sample}/I_{01}^{ref}}{I_{01}^{ref}/I_{00}^{ref}}\right|\right] \tag{3.26}$$

The two dark-field images can be added in quadrature as:

$$dark-field = |dark-field(vertical) + i dark-field(horizontal)| \tag{3.27}$$

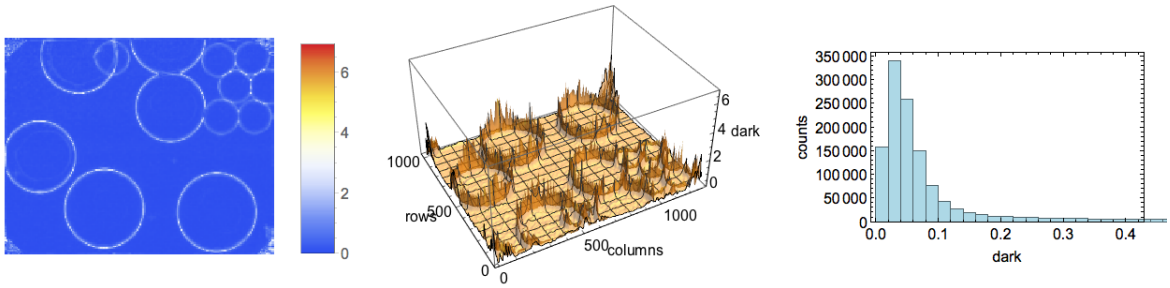


Figure 3.42: The dark-field image after adding in quadrature the horizontal and vertical images. Dark-field images are very good for edge and crack detection. The anamolous signal in the upper-left corner is where the grating structure ends; there is not real dark-field signal from this area. The numerical values of single-shot dark-field are roughly (1-dark-field) from a stepped-grating experiment. In the single-shot experiment, air=0 while in stepped-grating, air=1.

3.4.10 Differential Phase Contrast (DPC)

The differential phase contrast images are calculated as;

$$DPC(vertical) = \arctan\left[Re\left[\frac{I_{10}^{sample}}{I_{10}^{ref}}\right], Im\left[\frac{I_{10}^{sample}}{I_{10}^{ref}}\right]\right] \tag{3.28}$$

$$DPC(horizontal) = \arctan\left[Re\left[\frac{I_{01}^{sample}}{I_{01}^{ref}}\right], Im\left[\frac{I_{01}^{sample}}{I_{01}^{ref}}\right]\right] \tag{3.29}$$

The DPC is offset-corrected by subtraction of the mean.

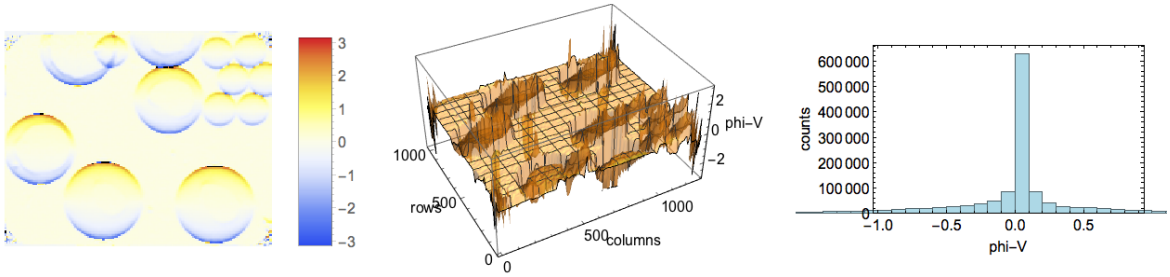


Figure 3.43: The vertical DPC calculated from the vertical harmonics. The DPC is offset-corrected by subtraction of the mean. (left) Some phase wrap is seen as white/black patches at the top and bottom edges of the spheres or (middle) as extreme up-down transitions. (right) Thehistogram is cropped, else values at $\pm\pi$ would be seen.

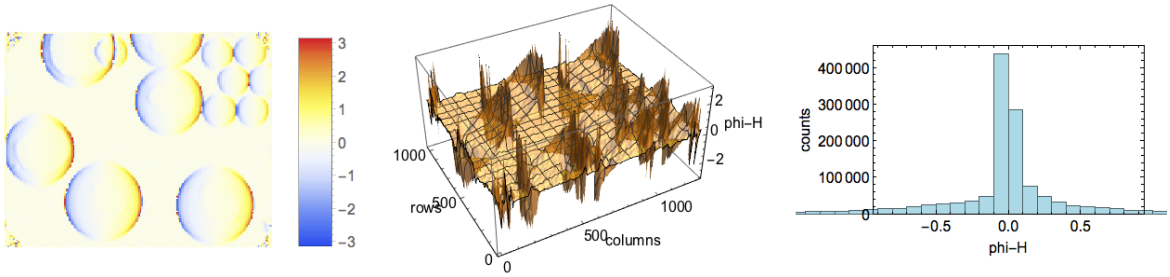


Figure 3.44: The horizontal DPC calculated from the horizontal harmonics. Again, some phase wrap is seen at the left and right edges of the spheres.

3.4.11 Phase Unwrapping: Haas

Phase unwrapping is a large topic with many options. Grating-based interferometry has the unusual advantage of an absorption image correlated with the phase image. Thus, a normalized derivative of the absorption image can be used to develop a cost function to

guide phase unwrapping.[19]

$$\mathcal{L}(U) = \sum_{(i,j)\in\{(i,j)|M_{ij}=1\}} \left(\alpha\Delta_x A_{ij} - [D_{ij} + 2\pi U_i j]\right)^2 \tag{3.30}$$

$$\alpha = \left| \frac{max(D)}{max(\Delta_x A)} \right| \tag{3.31}$$

3.4.12 Phase Integration: Frankot-Chellapa

Phase integration is really tricky. Good luck.

$$\Phi(x,y) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left[\Phi_x + i\Phi_y\right](k,l)}{2\pi i(k+il)}\right](x,y) \tag{3.32}$$

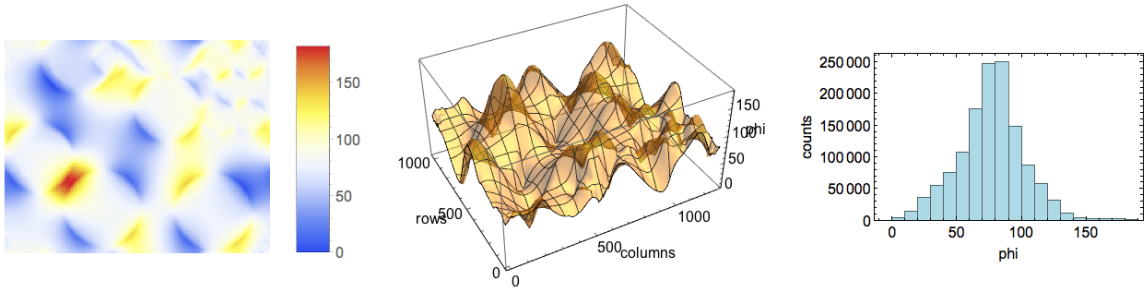


Figure 3.45: Attempted phase integration of Figs. 3.43 and 3.44, after processed with the Haas algorithm to remove some phase wrap. The result of this attempt Frankot-Chellapa phase integration is unacceptable.

3.4.13 Phase Integration: Shift in the Reciprocal Space

The integration of the two differential phases Φ_x and Φ_y are proposed in [20] and [21], and it is given by the Frankot-Chellapa equation:

$$\Phi(x,y) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left[\Phi_x + i\Phi_y\right](k,l)}{2\pi i(k+il)}\right](x,y). \tag{3.33}$$

This equation has the inconvenience of a singularity at zero frequencies when $k = l = 0$. Some authors have manually set a value for the term inside the IFT[8] at $k = l = 0$ [21],

[8]This is doing by noting that for $G(f_x) = \mathcal{F}[g(x)]$, we have that $G(f_x = 0) = \int_{-\infty}^{\infty} g(x)dx$

while others have added a small value ϵ to the denominator in order to avoid the division by zero.

The way used here is based in shifting the functions in the reciprocal space in order to avoid the singularity. Note that due to the properties of the discrete Fourier transform, any shift is in fact a *circular shift*[9]. This means that any shift that is a integer multiple of the bin Δk will contain the zero frequency. Therefore, to avoid the zero frequency we need to apply a shift of a fraction of the bin size.

One additional problem has to do with the FFT algorithm. It seems that, to calculate the FFT, all programs [10] require the value of the functions at $x = 0$ or at $k = 0$ (for FFT or for IFFT, respectively). Thus, even tough we can make a shift of the function we need to keep in mind that the FFT algorithm will consider the first element of the vector to be related to $x = 0$ or $k = 0$, the zero coordinates.

Below is shown how we address these problems. For the sake of clarity we start with the one dimensional case. The method to solve equation 3.33 is based in two properties of the Fourier transform, namely:

$$\mathcal{F}\left[\frac{\partial g(x)}{\partial x}\right](k) = 2\pi ik\mathcal{F}\left[g(x)\right](k), \tag{3.34}$$

$$G'(k) = 2\pi ikG(k), \quad \text{Differentiation Property} \tag{3.35}$$

and

$$\mathcal{F}\left[g(x)\exp(i2\pi xk_o)\right](k) = G(k-k_o). \quad \text{Translation Property} \tag{3.36}$$

By changing the variable k to $\tilde{k} - k_o$ in 3.35 we obtain

$$G'(\tilde{k}-k_o) = 2\pi i(\tilde{k}-k_o) \times G(\tilde{k}-k_o), \tag{3.37}$$

[9] see `https://en.wikipedia.org/wiki/Discrete_Fourier_transform`

[10] Most programs require that the first element of the vector is the one for $x = 0$ or $k = 0$.

and applying the translation property of the Fourier transform results in

$$\mathcal{F}\left[\frac{\partial g(x)}{\partial x}\exp(2\pi x k_o)\right](\tilde{k}) = 2\pi i(\tilde{k}-k_o)\times\mathcal{F}\left[g(x)\exp(2\pi x k_o)\right](\tilde{k}). \tag{3.38}$$

Finally, isolating $g(x)$ results in

$$g(x) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left[\frac{\partial g(x)}{\partial x}\exp(2\pi x k_o)\right](\tilde{k})}{2\pi i(\tilde{k}-k_o)}\right]\times\exp(-2\pi x k_o). \tag{3.39}$$

Points to highlight:

1. Since k_o is a fraction of the bin Δk and the values are discrete, then the singularity is gone.

2. For the IFT, the new Fourier coordinate is now $\tilde{k}$, which include the value $\tilde{k}=0$ and thus it fulfills the requirement of including the zero frequency.

3. k_o can be any non-integer number.

In two dimension the properties above results in:

$$g(x,y) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left[\left(\frac{\partial g(x,y)}{\partial x}+i\frac{\partial g(x,y)}{\partial y}\right)\exp[2\pi i(xk_o+yl_o)]\right](\tilde{k},\tilde{l})}{2\pi i(\tilde{k}-k_o+i\tilde{l}-il_o)}\right]\times\exp[-2\pi i(xk_o+yl_o)]. \tag{3.40}$$

3.5 References

[1] Marathe, S., Assoufid, L., Xiao, X., Ham, K., Johnson, W.W., Butler, L.G. (2014) "improved algorithm for processing grating-based phase contrast interferometry image sets". Review of Scientific Instruments **85**(1): art. no. 013704.

[2] Grunzweig, C., Pfeiffer, F., Bunk, O., Donath, T., Kuhne, G., Frei, G., Dierolf, M., David, C. (2008) "design, fabrication, and characterization of diffraction gratings for neutron phase contrast imaging". Review of Scientific Instruments **79**(5): art. no. 053703.

[3] Marathe, S., Xiao, X., Wojcik, M.J., Divan, R., Butler, L.G., Ham, K., Fezzaa, K., Erdmann, M., Wen, H.H., Lee, W.K., Macrander, A.T., De Carlo, F., Mancini, D.C., Assoufid, L. (2012) "development of grating-based x-ray talbot interferometry at the advanced photon source". AIP Conference Proceedings **1466**(1): 249–254.

[4] Gruner, S.M., Tate, M.W., Eikenberry, E.F. (2002) "charge-coupled device area x-ray detectors". Review of Scientific Instruments **73**(8): 2815–2842.

[5] Eggl, E., Schleede, S., Bech, M., Achterhold, K., Loewen, R., Ruth, R.D., Pfeiffer, F. (2015) "x-ray phase-contrast tomography with a compact laser-driven synchrotron source". Proceedings of the National Academy of Sciences of the United States of America **112**(18): 5567–5572.

[6] Velroyen, A., Yaroshenko, A., Hahn, D., Fehringer, A., Tapfer, A., Muller, M., Noel, P.B., Pauwels, B., Sasov, A., Yildirim, A.O., Eickelberg, O., Hellbach, K., Auweter, S.D., Meinel, F.G., Reiser, M.F., Bech, M., Pfeiffer, F. (2015) "grating-based x-ray dark-field computed tomography of living mice". EBioMedicine **2**(10): 1500–6.

[7] Lauridsen, T., Willner, M., Bech, M., Pfeiffer, F., Feidenhans'l, R. (2015) "detection of sub-pixel fractures in x-ray dark-field tomography". Applied Physics a-Materials Science & Processing **121**(3): 1243–1250.

[8] Bayer, F., Zabler, S., Brendel, C., Pelzer, G., Rieger, J., Ritter, A., Weber, T., Michel, T., Anton, G. (2013) "projection angle dependence in grating-based x-ray dark-field imaging of ordered structures". Optics Express **21**(17): 19922–19933.

[9] Revol, V., Kottler, C., Kaufmann, R., Neels, A., Dommann, A. (2012) "orientation-selective x-ray dark field imaging of ordered systems". Journal of Applied Physics **112**(11): art. no. 114903.

[10] Betz, B., Harti, R.P., Strobl, M., Hovind, J., Kaestner, A., Lehmann, E., Van Swygenhoven, H., Gruenzweig, C. (2015) "quantification of the sensitivity range in neutron dark-field imaging". Review of Scientific Instruments **86**(12).

[11] Strobl, M. (2014) "general solution for quantitative dark-field contrast imaging with grating interferometers". Scientific Reports **4**: art. no. 7243.

[12] Lynch, S.K., Pai, V., Auxier, J., Stein, A.F., Bennett, E.E., Kemble, C.K., Xiao, X., Lee, W.K., Morgan, N.Y., Wen, H.H. (2011) "interpretation of dark-field contrast and particle-size selectivity in grating interferometers". Applied Optics **50**(22): 4310–4319.

[13] Wen, H.H., Bennett, E.E., Kopace, R., Stein, A.F., Pai, V. (2010) "single-shot x-ray differential phase-contrast and diffraction imaging using two-dimensional transmission gratings". Optics Letters **35**(12): 1932–1934.

[14] Bennett, E.E., Kopace, R., Stein, A.F., Wen, H. (2010) "a grating-based single-shot x-ray phase contrast and diffraction method for in vivo imaging". Medical Physics **37**(11): 6047–6054.

[15] Itoh, H., Nagai, K., Sato, G., Yamaguchi, K., Nakamura, T., Kondoh, T., Ouchi, C., Teshima, T., Setomoto, Y., Den, T. (2011) "two-dimensional grating-based x-ray phase-contrast imaging using fourier transform phase retrieval". Optics Express **19**(4): 3339–3346.

[16] Takeda, M., Ina, H., Kobayashi, S. (1982) "fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry fourier-transformation method of fringe-patttern analysis for computer-based topography and interferometry". Journal of the Optical Society of America **72**(1): 156–160.

[17] Zanette, I., Weitkamp, T., Donath, T., Rutishauser, S., David, C. (2010) "two-dimensional x-ray grating interferometer". Physical Review Letters **105**(24).

[18] Olatinwo, M.B., Ham, K., McCarney, J., Marathe, S., J., G., Knapp, G., Butler, L.G. (2016) "analysis of flame retardancy in polymer blends by synchrotron xray k-edge tomography and interferometric phase contrast movies". J. Phys. Chem. B **120**: 2612–24.

[19] Haas, W., Bech, M., Bartl, P., Bayer, F., Ritter, A., Weber, T., Pelzer, G., Willner, M., Achterhold, K., Durst, J., Michel, T., Prmmer, M., Pfeiffer, F., Anton, G., Hornegger, J.: Phase-unwrapping of differential phase-contrast data using attenuation information. In: Proceedings of SPIE. Volume 7962., SPIE 79624R–79624R–6 10.1117/12.877945.

[20] Kottler, C., David, C., Pfeiffer, F., Bunk, O. (2007) "a two-directional approach for grating based differential phase contrast imaging using hard x-rays". Optics Express **15**(3): 1175–1181.

[21] Morgan, K.S., Paganin, D.M., Siu, K.K.W. (Sep 2011) Quantitative single-exposure x-ray phase contrast imaging using a single attenuation grid. Opt. Express **19**(20): 1978119789.

Chapter 4
Experiments

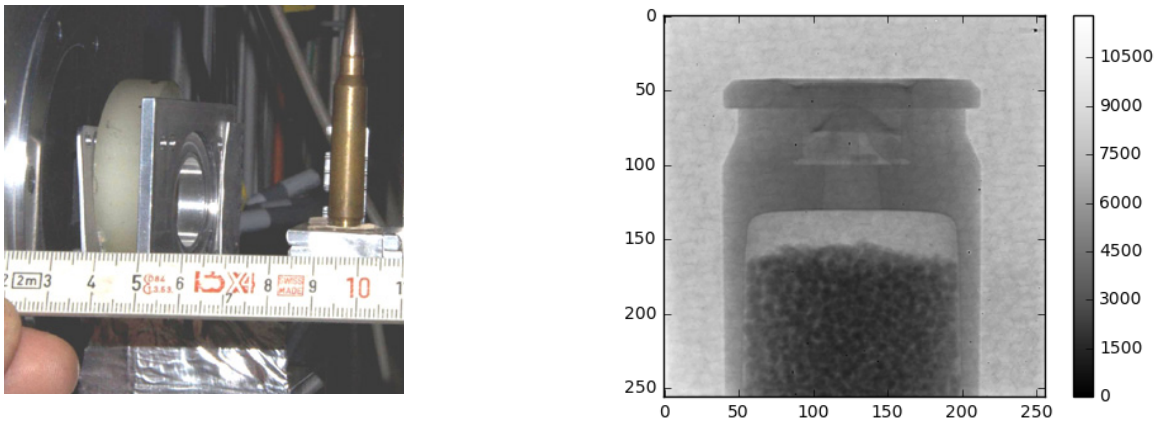
4.1 Neutron Tomography: Bullet

The neutron tomography was conducted at the Paul Scherrer Institute ICON beamline. The neutron source is provided by nuclear spallation of a heavy metal target irradiated by a proton beam. The emitted neutrons are slowed by a liquid hydrogen moderator giving a neutron beam with a useful wavelength range of 1 to 4.5 Å. The neutron beam was collimated with a 2 cm diameter (D) pinhole 7 m (L) upstream of the sample giving a L/D ratio of 350. The neutron flux was 8×10^6 n/cm^2/s and stable over the time of the experiment. The sample was mounted on a rotation stage 5 cm from the detector. With this configuration, the assumption of parallel beam tomography is valid to a geometric blur of 140 μm.

A novel neutron imaging detector was tested with a variety of samples, including a bullet. The detector was based on neutron capture by ^{10}B, then measurement of the ejected electrons with a microchannel plate and Timepix detector. The Timepix is a hybrid semiconductor with 256×256 pixels, each 55 μm square, connected to energy discriminators counting electronics. The detector is linear over the neutron flux, free of optical distortion, and has negligible dark count. As a new technology, the bump-bonding of pixels to their respective energy discriminator circuitry was nearly perfect, though a few pixels were inoperable. The inoperable pixels lead to divide by zero faults in the uncorrected data analysis. Also, readout failures occasionally lead to anomalous images. The workflow must be flexible to account for realistic data.

The tomography data was collected with 150 s exposures and 201 projections over 180°. The image files are text. A reference image with sample removed from the beam (white

field) was collected with 600 s exposure; an image with neutron beam off (dark field) was collected with 100 s exposure.



(a) (b)

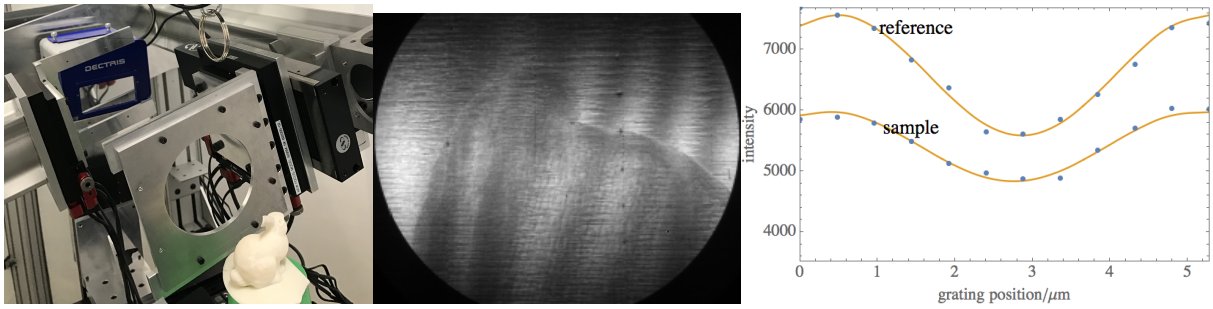
Figure 4.1: (a) The bullet and the neutron detector, visible above the tape measure at 4 to 5 cm. Later, the bullet was remounted with the primer end up. (b) A raw image with colorbar in neutron counts.

4.2 X-ray Tomography: Bunny

X-ray interferometry provides a dark-field image, essentially a small-angle X-ray scattering image, of the voids and print defects in an additively manufactured polymer object. The X-ray tomography/interferometry experiments were performed at the LSU CAMD synchrotron tomography beamline. The tomography beamline was use as a prototype for a stepped-grating Talbot-Lau interferometer operating with a microfocus X-ray tube source. In grating interferometry, three X-ray optics, gratings, are positioned around the sample and one grating is moved incrementally in sub-micron steps, often on the order of twelve steps. The result is a set of images that form an interferogram. A least squares solution of the interferogram yields three images of the sample: absorption, differential phase contrast, and dark-field (similar to a small angle scattering image). Like the neutron tomography experiment, interferograms are collected with the sample as a function of tomography rotation angle as well as reference interferograms collected before and during the sample collection.

The X-ray detector was a Pilatus 100K photon counting system. Like the neutron work, the Pilatus is free from optical distortion, is linear with flux, and has negligible dark counts. There are a few non-functioning pixels. The Pilatus 100K has square 172 μm pixels formatted as 487 rows and 195 columns. The optics holders limited the field-of-view to a round disk with a diameter of 230 pixels. In this work, the tomography experiment was repeated with the interferometry operating in two modes, one for imaging along the laboratory vertical axis and another for imaging along the laboratory horizontal axis.

The sample was a 3D printed Stanford Bunny printed with 1.75 mm diameter acrylonitrile butadiene styrene (ABS) filament. The Stratasys Dimension Elite was set to a layer thickness of 0.254 mm. The research question was the quality of the filament-to-filament bonding, i.e., the presence of anisotropic sub-micron cracks and voids that are visible in the dark-field image for one interferometer configuration, but not the other. To define the role of the tomography rotation axis, experiments were performed with the Stanford Bunny in "feet down" and "nose down" orientations on the tomography rotation stage.



(a) (b) (c)

Figure 4.2: (a) An ABS 3D printed Stanford Bunny mounted "feet down" on a tomography rotation stage in a Talbot-Lau interferometer (gratings removed for photography). One ear of the Bunny was removed for SEM imaging of the filament-to-filament sub-micron cracks. (b) Raw image of the "nose down" orientation. The gratings are aligned along the lab horizontal (or vertical). (c) The data points were extracted from the center of the twelve sample and reference images. The fits were made with a least square algorithm [Marathe2014].

The absorption and dark-field volumes are used to correlate printhead trajectory with print defect density. The absorption volume is used to generate perimeter points slice-by-

slice, and from these points, the 2D curvature is calculated. There is a slight increase in X-ray scattering, hence print defect density, at regions with high curvature.

Two X-ray interferometry techniques were used: stepped-grating and single-shot. As currently developed, stepped-grating has the larger field-of-view—examination of an entire test object—whilst single-shot has the potential for real-time, *in situ* measurement of the printing process within 1 mm of the printhead.

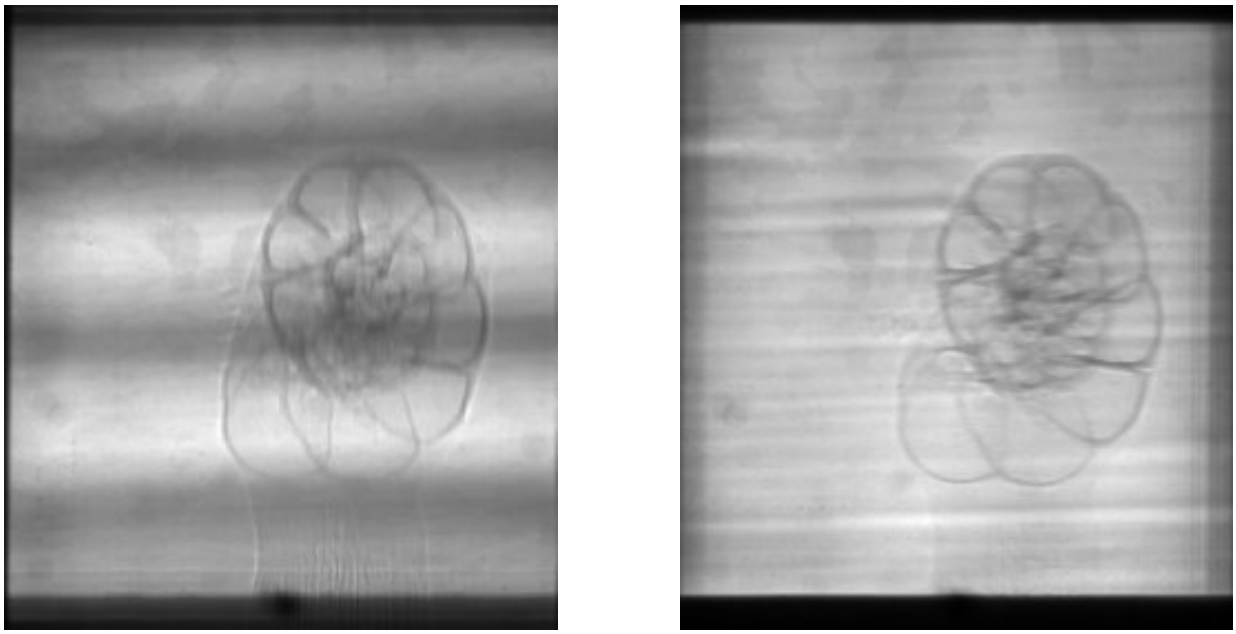
4.3 Formanifera

The first experimental example is an interferogram of small biological sample, a foraminifera, a one-cell, ocean-dwelling protist. A raw image dimly shows the foram affixed to a wooden toothpick, Fig. 3.7. The striations in the background are attributed to support structures in the grating.

Raw image of the mostly calcium carbonate shell of a foraminifera, about 6 mm across. The colorbar gives the CCD counts. The gratings are aligned along the lab horizontal to take advantage of the vertical phase coherence of the beamline, while The gratings are aligned along the lab vertical to take advantage of the horizontal phase coherence of the beamline.

4.4 Dogbone

Recently, tensile stressed SS315 dogbones created with sintered laser melting were studied with neutron grating-based interferometry/tomography. A combination of two imaging modalities from the experiment, attenuation and dark-field, were processed to generate line intensity profiles of normalized neutron scattering versus position along the dogbone. In the 75% stressed sample, regions of interest were clearly visible. Post-imaging, the sample was subjected to additional stress to failure; the fracture point corresponded to a region of interest [1]. Post-fracture, SEM and EBSD microscopy supports a correlation of dark-field image sensitivity to sub-micron porosity evolution at stressed regions in the SLM AM dogbone [2]. The ability to non-destructively observe crack formation in tensile stress AM dogbones is likely to lead to *in situ* test jigs for real time tensile testing and



(a) (b)

Figure 4.3: A raw image of foraminifera in (a) horizontal grating and (b) vertical grating.

interferometry, similar to previous work with Bragg edge imaging [3].

Admittedly, neutron interferometry/tomography facilities are extremely scarce, with only a handful of facilities world-wide: NIST, HZB, PSI, FRM II, and J-PARC. However, as the optimal scattering length is determined [4], the potential grows for application of laboratory X-ray systems for non-destructive evaluation with grating interferometry. Hence, the time is ripe for interferometry/tomography reconstruction with advanced beamlines.

The neutron interferometry experiments were performed with two systems. The Talbot-Lau experiments were done at Helmholtz Zentrum Berlin on the CONRAD2 beamline.[5] The far-field experiments were done at the NIST Neutron Imaging Facility on the CG-6 beamline.[6] Recent work with Talbot-Lau on titanium samples made with electron beam melted additive manufacturing have shown that a correlation of dark-field with attenuation imaging highlights imperfections in the printed sample.[1]

Fig. 4.4 shows three SS316 sintered laser melting (SLM) additive manufacturing samples positioned on a tomography stage immediately upstream of the G1 grating of a Talbot-Lau neutron interferometer. The interferometer setup and the sample position

combine to make the dark-field image most sensitive to scattering centers near 2 μm in size, ξ_{setup}=1.97 μm, complementing the attenuation effective pixel size of 59 μm.

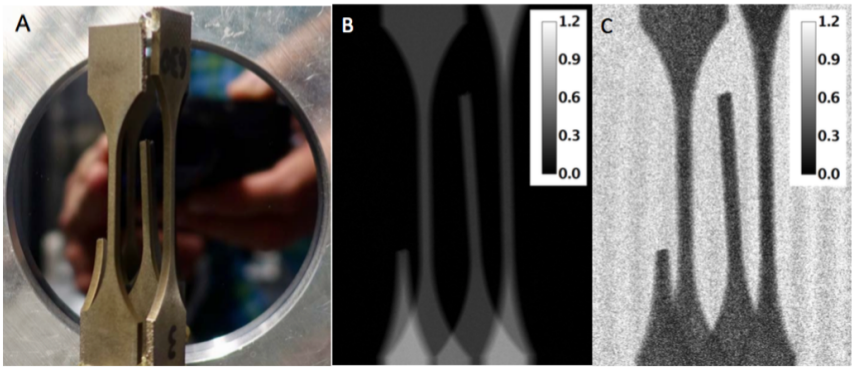


Figure 4.4: (A) Three SS316 dogbones as imaged at the Helmholtz Zentrum Berlin neutron imaging beamline CONRAD2 beamline. Behind the dogbones is a silicon wafer with micro-fabricated neutron optics for the interferometry experiment. The stepped-grating neutron interferometry experiment yields (B) attenuation and (C) scattering (dark-field) images.

The optical and projection images in Fig. 4.4 show three AM SS316 dogbone arranged in a square pattern on the tomography rotation stage; the flat tomography stage provides a reference plane for distance measurements along the lengths of the dogbones. All dogbones were originally 80.0 mm long. The dogbones were examined in three states: pristine, tensile stressed to 75% to failure, and stressed to fracture. The 75% stressed sample was imaged at 83.6 mm length. The two fractured pieces are 31.4 mm and 54.6 mm. The 75% stressed sample is directly opposite the short fractured sample. Together, the short and long fractured samples provide distance reference markers for the other two samples.

4.5 References

[1] Brooks, A.J., Ge, J., Kirka, M.M., Dehoff, R.R., Bilheux, H.Z., Kardjilov, N., Manke, I., Butler, L.G. (2017) Porosity detection in electron beam melted ti-6al-4v using high-resolution neutron imaging and grating-based interferometry. Progress in Additive Manufacturing **2**(3): 125–132.

[2] Weafer, F., Guo, Y., Bruzzi, M. (2016) The effect of crystallographic texture on stress-induced martensitic transformation in niti: A computational analysis. Journal of the Mechanical Behavior of Biomedical Materials **53**(Supplement C): 210 – 217.

[3] Woracek, R., Penumadu, D., Kardjilov, N., Hilger, A., Boin, M., Banhart, J., Manke", I. (2015) Neutron bragg edge tomography for phase mapping. Physics Procedia **69**: 227

– 236. Proceedings of the 10th World Conference on Neutron Radiography (WCNR-10) Grindelwald, Switzerland October 510, 2014.

[4] Malecki, A., Eggl, E., Schaff, F., Potdevin, G., Baum, T., Garcia, E.G., Bauer, J.S., Pfeiffer, F. (2014) Correlation of x-ray dark-field radiography to mechanical sample properties. Microscopy and Microanalysis **20**(5): 1528–1533.

[5] Manke, I., Kardjilov, N., Schafer, R., Hilger, A., Strobl, M., Dawson, M., Grunzweig, C., Behr, G., Hentschel, M., David, C., Kupsch, A., Lange, A., Banhart, J. (2010) Three-dimensional imaging of magnetic domains. Nature Communications **1**: art. no. 125.

[6] Pushin, D.A., Sarenac, D., Hussey, D.S., Miao, H., Arif, M., Cory, D.G., Huber, M.G., Jacobson, D.L., LaManna, J.M., Parker, J.D., Shinohara, T., Ueno, W., Wen, H. (2017) Far-field interference of a neutron white beam and the applications to noninvasive phase contrast imaging. Phys. Rev. A **95**: art. no. 043637.

Chapter 5
Scientific Workflows

Scientific Workflows play an important role for computational experiments in additive manufacturing 3D printing and interferometry/tomography imaging analysis. A clear workflow template allows scientists to process experiments easier and faster. Workflow library grows, but to find an appropriate workflow for their task is challenging.

In Garijo's paper [1], the authors have manually analyzed over 260 workflows encoded in open source and commercial software to extract common features. The results show two major motifs, data input/output and calculations, which are described by a motif ontology subdivided into categories such as "data cleaning", "atomic workflows", and "human interaction". Data cleaning and human interaction are well known by the tomographer; atomic workflow refers to a hands-off computation such as submitting a list of cleaned and centered sinograms for slice reconstruction. Therefore, we use the motif ontology, including color coding of flowcharts, of Garijo et al. to describe workflows and apply the workflow template to our sample datasets.

Workflows for interferometry/tomography will be presented herein as a progression from the development workflow to the finished product. In our lab, we use Mathematica workflows for development of new analysis or the incorporation of new equipment into the experiment. One example of a Mathematica workflow describes optimization of chemical engineering simulations with an emphasis on detecting sensitive parameters or control points.[2] Mathematica notebooks are converted to Jupyter/TomoPy/ASTRA notebooks which have been highly successful for processing interferometry/tomography data sets by new and relatively inexperienced tomographers. In its brief history, Jupyter has already been employed to extract data from a geology database and perform analysis from a range

of over 400 established programs and functions.[3]

5.1 TomoPy/ASTRA/Jupyter workflow

5.1.1 Introduction

TomoPy (version 1.1.0)[1] is an open-source Python package for tomography. TomoPy provides reconstruction algorithms for tomography, filters, ring removal algorithms, phase retrieval algorithms, and forward projection operator for absorption and wave propagation [4], including a parallel tomography reconstruction code, GridRec. TomoPy was developed to address the needs for tomographic reconstruction in an instrument-independent manner, primarily at synchrotron beamlines but also for users of neutron beamlines and laboratory X-ray instruments.

In our laboratory, the features we use are:

- ASTRA [2] contains a suite of reconstruction algorithms including filtered back projection (FBP) and simultaneous iterative reconstruction technique (SIRT), and validated by the Vision Laboratory, University of Antwerp. ASTRA supports parallel and fan beam geometries.[5, 6, 7]

- Tomopy supports an open source development community through Github. In fact, we have contributed our interferometry codes to TomoPy. Other open source developers have contributed noteworthy codes such as ring artifact removal.

- The TomoPy modules, while accessible through a python session in a text-based terminal, is also compatible with a graphical interactive notebook environment such as Jupyter. These features promote rapid development and provenance, respectively.

5.1.2 Workflow

For interferogram step, we have already elaborated in Chapter 3. Here, we only focus on reconstruction part in Jupyter notebook. In the first step, we import TomoPy packages

[1] `https://tomopy.readthedocs.io/en/latest/`
[2] `http://www.astra-toolbox.com/`

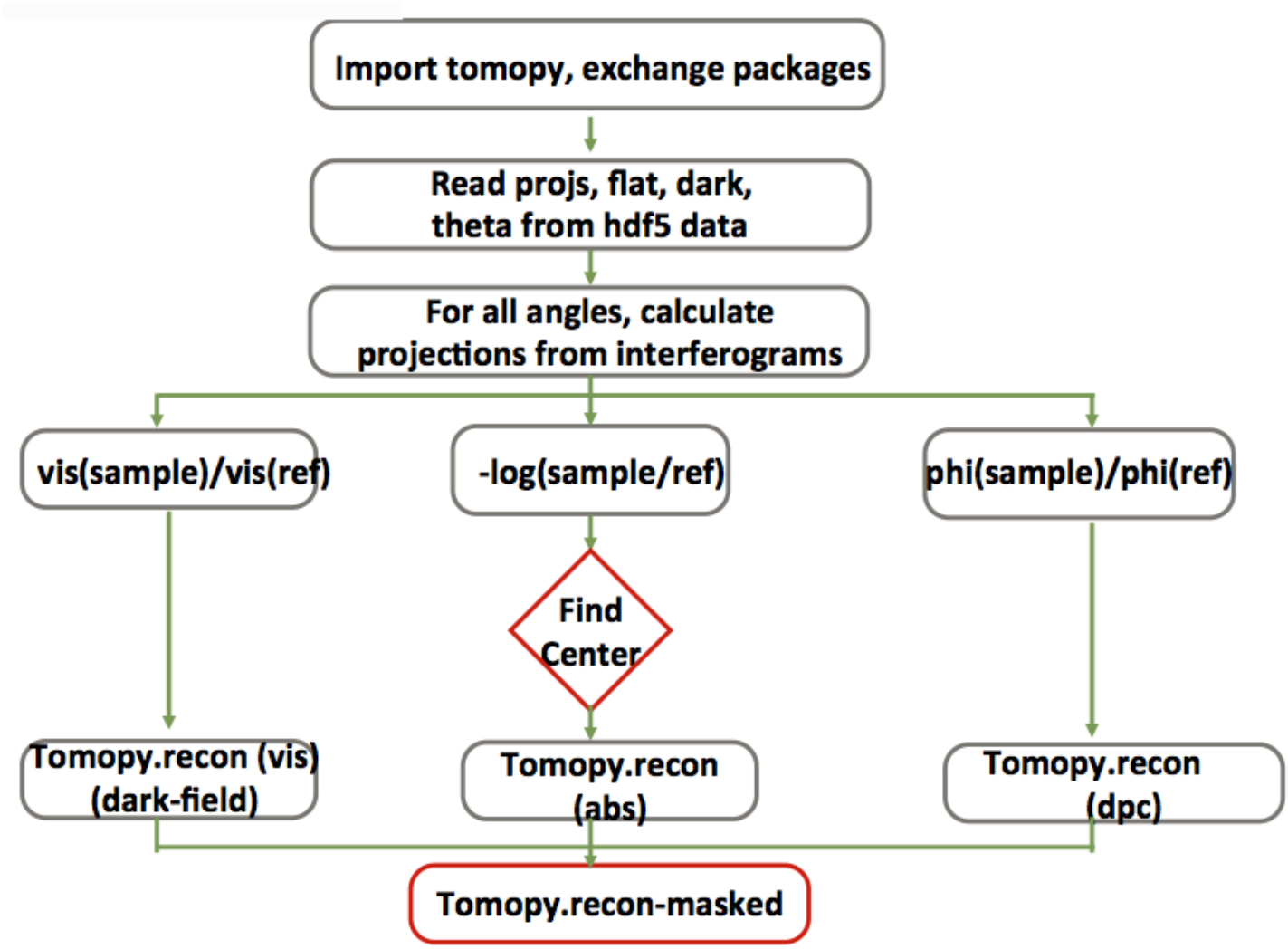


Figure 5.1: A generic flow chart of the TomoPy/ASTRA/Jupyter workflow for processing raw images into reconstructed absorption, differential phase contrast, and dark-field (scattering) volumes. Based on the scientific workflow motifs in Garijo et al. paper [1]

[3]; read projections for absorption, dark-field, DPC from stepped-grating interferometry analysis. 'Theta' here means all angles from interferograms. Centering for image reconstruction could be determined through absorption volumes due to clear contrast between sample and air with command,

```
rot_center = tomopy.find_center(proj, theta, init=236, ind=28, tol=0.5)
```

Sometimes, the rotation center value might not be accurate. Users can redefine the value based on the quality of reconstructed images.

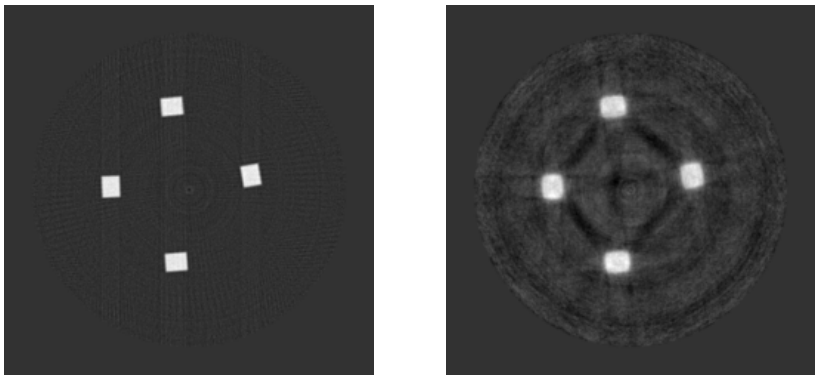
[3] https://tomopy.readthedocs.io/en/latest/release/notes-1.0.0.html

In TomoPy reconstruction libraries, there are a few options, like 'art' (Algebraic reconstruction technique), 'fbp' (Filtered back-projection algorithm), 'gridrec' (Fourier grid reconstruction algorithm) etc. During these method, we eventually found 'gridrec' generate a clear mask and object structure.

```
recon = tomopy.recon(absProj, theta, center=rot_center, algorithm='gridrec')
recon = tomopy.circ_mask(recon, axis=0, ratio=0.75)
```

For dark-field image reconstruction, after a extensive investigation of reconstruction methods, 'SIRT' in ASTRA toolbox denotes a satisfied dark-field volume.

```
extra_options ={'MinConstraint':0}
options = {'proj_type':'cuda', 'method':'SIRT_CUDA',
'num_iter':40,'extra_options':extra_options}
darkRecon = tomopy.recon(darkProj, theta, center=rot_center,
algorithm=tomopy.astra, options=options)
darkRecon = tomopy.circ_mask(darkRecon, axis=0, ratio=0.75)
```



(a) (b)

Figure 5.2: Reconstructed slices in Jupyter notebook with the methods of (a) 'Gridrec' in TomoPy for absorption images and (b) 'SIRT' in ASTRA for dark-field images

5.2 Vistrails Workflow

5.2.1 Pioneer Work: SNARK14

Since VisTrails and TomoPy were not released before 2014, at that time, we used another software for sample reconstruction, SNARK09 and SNARK14 [4], where SNARK14 is an updated version of SNARK09. Executed in Linux system, SNARK09 provides a viable image reconstruction in several years ago. During that period, software for image reconstruction was not extensive. Generally, the features in SNARK09 are prominent [8]

- Polychromatic and monochromatic X-ray simulation
- Beam hardening correction
- Projection computation
- Digital difference analyzer (DDA)
- Basis functions
- Reconstruction algorithms

However, SNARK09 only works on Linux system and the data format in 'input' and 'output' files are not easy to be modified. Meanwhile, some controls are inefficient and inflexible. For example, there is no option to user define the parameter 'rotation center' and the manipulation relies on Linux command lines through terminal. Since then, Garbor Herman at CUNY (the City University of New York) planned to rewrite some of the existing code so as to make it computationally more effectively and faster. Herein, SNARK14 was released five years after SNARK09. In this new version, reconstruction methods were highly improved, a GUI was developed for non-programming users and more controls in SNARK14. Furthermore, a virtual box with SNARK14 installed is available online [5]. More

[4] `http://turing.iimas.unam.mx/SNARK14M/`

[5] `http://turing.iimas.unam.mx/SNARK14M/documentation.php`

features came up in SNARK14. However, we still on the way looking for a robust software for interferometry image reconstruction without using virtual box in Linux system.

5.2.2 Introduction

The tomography software landscape is rapidly evolving towards an emphasis on workflows. Tomography software is available as a single package for raw images to reconstructed volumes; MuhRec is a current example of a package with an extensive GUI for cropping and center of rotation correction [9]. The SNARK series allows exploration of a wide range of reconstruction algorithms [10]. High performance X-ray synchrotron beamlines with high data rates—0.5 TB/minute!— have built custom packages; the Swiss Light Source TOMCAT facility developed a Python package and report cost/benefit analysis for bone imaging [11, 12]. Researchers in the iMinds-Vision Lab at the University of Antwerp have posted the ASTRA library, a GPU-enable set of tomography reconstruction codes ready for integration into Python or Matlab workflows.[5, 6] Recently, the TomoPy library [4], developed in part at the Advanced Photon Source, has incorporated the ASTRA library.[7]

At present, VisTrails provides a reliable tomography GUI for non-programmers, which combines features of both workflow and visualization systems. VisTrails workflows are the pentultimate goal for our tomography projects. VisTrails offers a unique combination of visualization and data provenance. Data provenance gives the tomographer a history of the processing steps performed by the VisTrails workflow, i.e., a unique tomography reconstruction is logged and can be repeated in the future. VisTrails is an open source workflow system currently maintained by Prof. Juliana Freire and co-workers at New York University.[13, 14] VisTrails has been used by researchers for projecting spanning wildfire simulations [14] to public domain development of supernova simulations [15] For interferometry/tomography, one simple VisTrails workflow has been developed and is presented herein. The challenges have include incorporation of human interaction and barrier to updates from the Mathematica and Jupyter/TomoPy/ASTRA workflows.

5.2.3 Mathematica Workflow

In our lab, Mathematica notebooks have been used for tomography data processing workflows for nearly ten years. In the nomenclature of Garijo *et al.*, Table 4 [1], the notebooks have been developed for the varied problems of data preparation—group, sort, format—followed by data cleaning and visualization. Furthermore, the notebooks contain sections that can be described as atomic workflows—interferometry algorithm and tomography reconstruction—and sections which require human intervention. Graphs and plots are generated to confirm or solicit input on interferometry processing, reference image grouping, center of rotation, and binarization threshold.

A small tomography project is fully contained in the Mathematica notebook and associated text files for the Bullet project [6]. The notebook structure roughly follows the flowchart in Fig. 6.11. In Step 1, some functions are defined; Step 2 reads image filenames to determine image sequence and grouping—human intervention is required to repair two images; Step 3 calculates the absorption projections as a function of rotation angle. As this is not an interferometry experiment, the visibility and phi projections do not exist; Step 4 finds the center of rotation—human intervention is needed to select the best rotation center; Step 5 is an atomic workflow for tomography reconstruction using the Mathematica version of filtered backprojection; and Step 6 is mask generation and visualization of the reconstructed volume. The Mathematica notebook and dataset are distributed; please acknowledge the Paul Scherrer Institue, Neutron Imaging and Activation Group [7] where the tomography data was collected.

Larger projects with interferometry data used a somewhat expanded Mathematic notebook, but the essential features remain. The notebooks do make use of high performance computing by running on a single node, multi-core system using the remote kernal option. Of all the workflow software in hand, Mathematica offers the most convenient multiplaform and HPC operation. The continued evolution of the tomography workflows starts with the

[6] `https://github.com/lsu-lesbutler/LSU-VisTrails`
[7] `https://www.psi.ch/niag/`

Mathematica notebooks due mainly to the upgrades within the Mathematica language. We can point to the inclusion of image processing commands such as a total variation filter [16] and inpainting [17], both developed within the last few years and reliably included in Mathematica.

5.2.4 Jupyter Workflows

A Jupyter notebook is an open-source, server-client application derived from iPython [8]. Jupyter allows programming through a web browser and has support for over 40 programming languages. In our lab, remote access is through NoMachine. Equations, visualization, data cleaning, and many other functions are accessible through the Jupyter notebook. Herein, we apply use the iPython kernel in a Jupyter v5 notebook to execute Python scripts. In geochemistry, Jupyter notebooks are used to access and re-analyze archived paleomagnetic data from a consortium collection [3].

For the 3D printed samples, the TomoPy package was accessed through Jupyter. TomoPy [4] (Sec. 5.3.1) is an open-source Python package for tomographic processing and reconstruction with GridRec. Recently, TomoPy has incorporated the ASTRA toolbox [7], adding a number of 2D and 3D tomography reconstruction algorithms, including filtered backprojection (FBP) and simultaneous iterative reconstruction technique (SIRT), both GPU-enabled [5, 6].

Here, we conducted two experiments: a simple bullet data set for neutron imaging and two bunny data sets for X-ray grating interferometry. = As in the Mathematica workflow, the bullet project is fully contained in a Jupyter notebook [6]. The notebook structure roughly follows the flowchart in Fig. 6.11. The Juypter and Mathematica workflows are quite similar for Steps 1 (import) to 4 (find center), including human intervention for finding the center of rotation. Step 5 (reconstruction) is noteworthy; here, an atomic workflow is implemented with the TomoPy/ASTRA package for reconstruction using either GridRec or FBP. The tomography volumes, HDF5 and raw binary, are visualized in ImageJ and

[8] `http://jupyter.org`

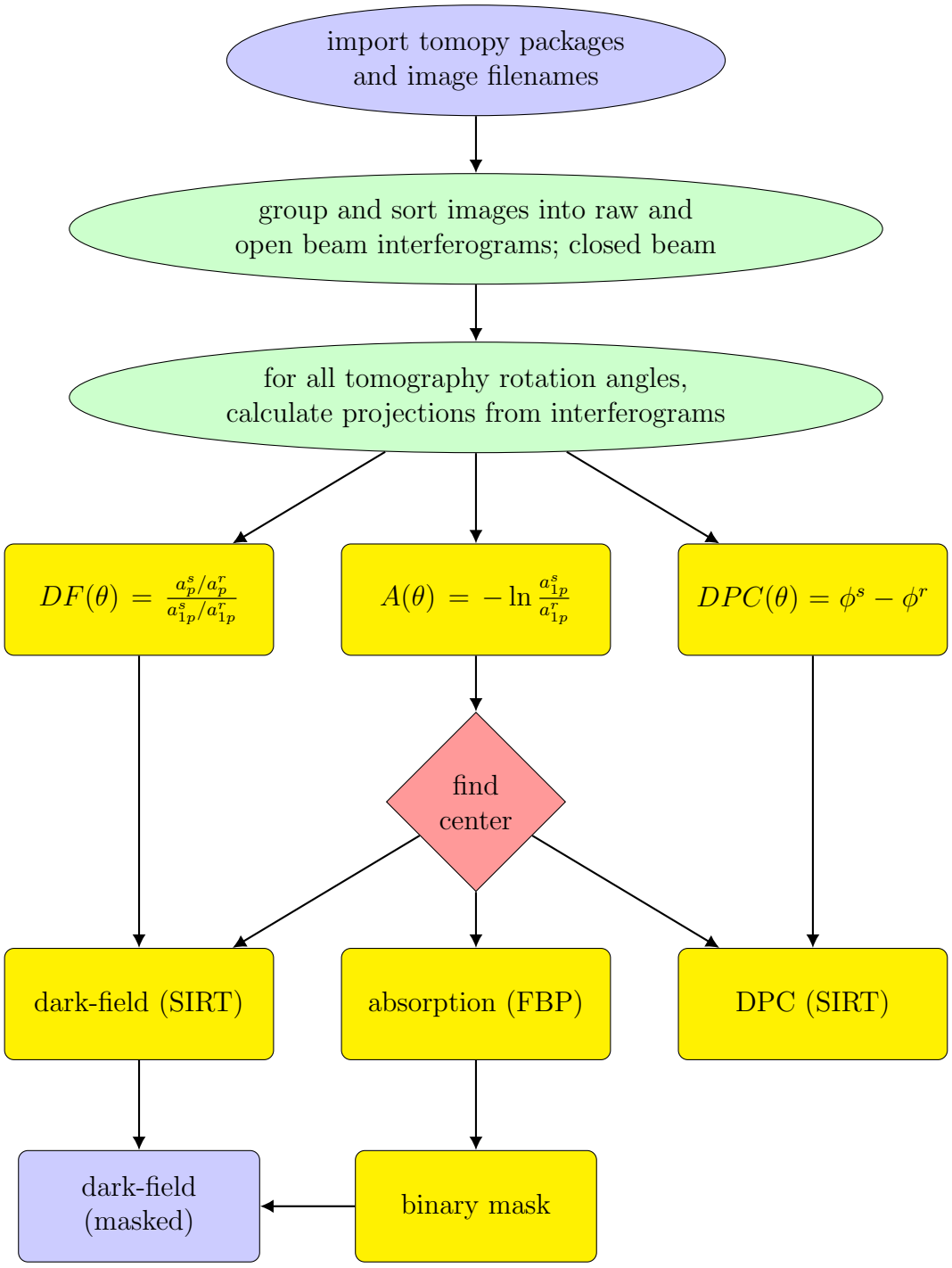


Figure 5.3: A generic flow chart of the workflow for processing raw images into reconstructed absorption, differential phase contrast, and dark-field (scattering) volumes. Based on the scientific workflow motifs in Garijo et al. paper [1], blue ellipse represents input augmentation; green ellipse represents data preparation; yellow rectangle represents data analysis; red diamond represents parameter adjustment and blue rectangle represents output extraction.

Avizo 7, respectively.

The bullet data set is small, containing only three image types—raw, open beam, closed beam—and ordered over a single parameter, the rotation angle. The next data sets again have raw, open beam, and closed beam images, but now ordered with respect to three parameters: grating step position x_g, rotation angle θ, and time (image index number). Time becomes important so as to account for interferometer drift during the experiment. The looping structure over image type, rotation angle, and grating step position can be viewed as shown in Scheme 1,

$$
\begin{aligned}
image\ index\ i &= 1 \\
For\ \theta &= [0, 180, \Delta\theta]\ degree \\
For\ type &= [raw,\ open\ beam,\ closed\ beam] \\
For\ x_g &= [1, n\ x_g, x_g]\ \mu m \\
&\quad If\ \theta = 0, image\ closed\ beam,\ i++ \\
&\quad If\ Mod[\theta, m\ \theta] = 0, image\ open\ beam\ interferogram,\ i++ \\
&\quad image\ raw\ interferogram,\ i++
\end{aligned}
$$

Scheme 1 : Pseudocode for image labeling of filenames

leading to images grouped and ordered as Image$(type, i, \theta, x_g)$. In the event of an experiment interruption, such as a synchrotron beam dump, experiment restart can be accomplished by backing up to the last completed raw interferogram, then reacquiring an open beam, then resuming raw interferogram collection. The workflow software then groups the raw interferograms with the most recent, complete open beam interferograms.

While, for our bunny data sets, we used X-ray grating interferometry instead of neutron imaging as for bullet. Two big data sets are fully contained in the Jupyter notebook and associated tiff files for the Bunny project [6]. The notebook structure roughly follows the same flowchart in Fig. 6.11. In Step 1, some functions are defined; especially, we grouped

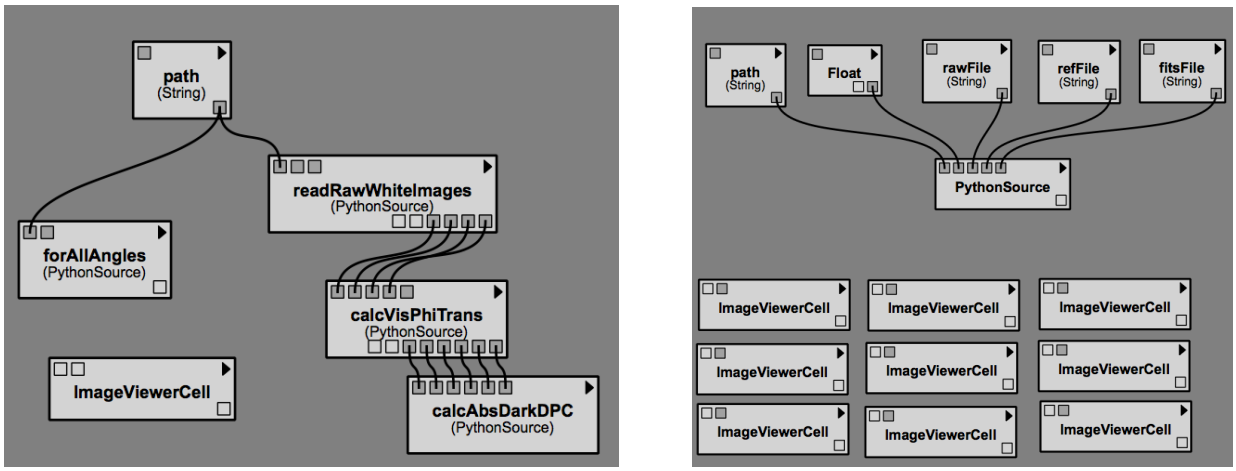
sample files and reference files according to the same angle; Step 2 reads raw tiff filenames to determine image sequence and grouping; In our interferometry experiment, there are 12 grating steps for each rotation angle. Step 3 calculates the transmission, visibility and phi for all reference and sample images based on vector calculations. Step 4 calculates the absorption, dark-field and differential phase contrast (DPC) for all slices based on the math in Fig. 6.11. Step 5 finds the best center of rotation—human intervention is needed to select the best rotation center in Jupyter notebook; Step 6 is an atomic workflow for tomography reconstruction using the TomoPy/ASTRA packages; "gridrec" in TomoPy package and "FBP" in ASTRA package work well for absorption reconstructions; meanwhile, GPU-accelerated algorithms "SIRT" and "SART" in ASTRA package give us satisfied results in dark-field and DPC reconstructions; and Step 7 is mask generation and visualization of the reconstructed volume with Avizo 7.

5.2.5 VisTrails Workflows

For non-programmers, VisTrails (Version 2.2.4) provides an alternative option for image reconstruction. We wrapped our Python scripts in Jupyter notebook as user-defined modules.

Two big data sets for the Bunny project [6] are fully contained in the VisTrails. The workflow thread roughly follows the same flowchart in Fig. 6.11. In Step 1, some functions and path strings are defined by users through the input module; Step 2 connects all function modules and path modules together; then apply to Python source module. The Python source module is hidden from users. Especially, some parameters need be defined by users in this Python source module. Step 3 executes this workflow; text output will be shown in terminal window, while images will be displayed in VisTrails spreadsheet. Here are the templates VisTrails workflows for bunny interferometry calculation and image reconstruction.

As shown in Fig. 5.5, for convenience, we split interferometry/tomography imaging analysis into two parts, As for the first part in (a), the **path** module is to define working



(a) (b)

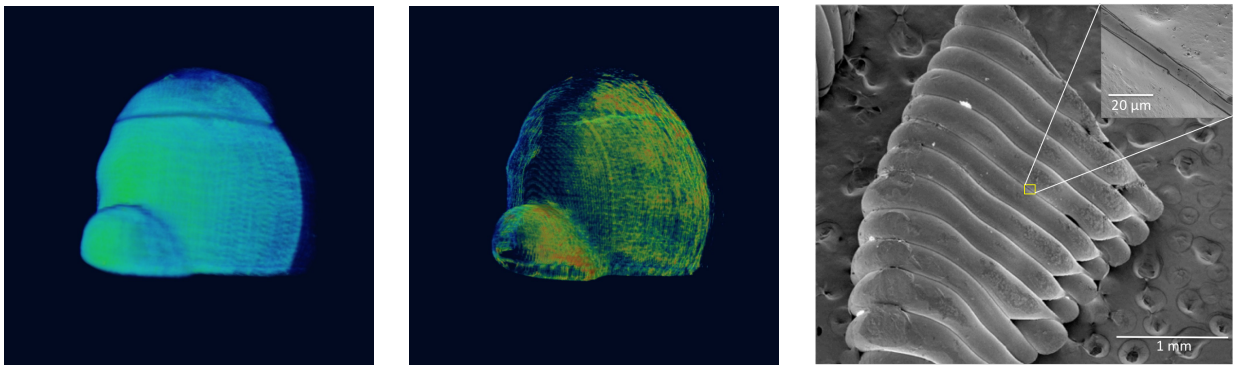
Figure 5.4: (a) vistrails workflow for bunny interferometry calculation and (b) VisTrails workflow for bunny reconstruction

directory. The left node in the workflow, **forAllAngles** is a parameter control module to input angle information. The right nodes in the workflow indicate grating-based interferometry analysis in Python source. **readRawWhiteImages** is reading raw files from **path**; **calcVisPhiTrans** is to calculate visibility, phi and transmission matrix with vectorized least square method discussed in Chapter 3; next, calculate absorption, dark-field and DPC projections from visibility, phi and transmission with mathematical equations (Chapter 3). **ImageViewerCell** enables 2D image view in VisTrails spreadsheet. For the second part (b), image reconstruction via TomoPy/ASTRA package, **path** is working directory; **Float** means user-defined parameters; **fitsFile** is where the exported fits projections located after grating-based interferometry analysis. All the initialization and user-defined parameters will eventually be imported into **PythonSource** – TomoPy/ASTRA/Jupter notebook for image reconstruction.

5.2.6 Results and Conclusion

We worked hard to find a way to convert well-tested Mathematica and Matlab codes into a VisTrails compatible structure. Discuss the reasons for DumpSave, installation of xvfb, nominally a X-server debugger tool. Now, we find an easier way to utilize TomoPy-/ASTRA packages in our Python scripts. Additionally, we use JupyterHub as our python editor for multi-users.

With Avizo 7, we acquired 3D volume for absorption and dark-field images. The dark-field images of 3D printed objects in Fig. 6.3b consistently showed voids and print defects between filament layers in ABS (Acrylonitrile Butadiene Styrene) and PLA (Polylactic Acid) objects, as demonstrated here for two Stanford Bunnies and a quadratic test object. The scattering affects the dark-field imaging modality with a sensitivity dependent upon the interferometry parameters; the optical elements are rotated to detect scattering parallel and perpendicular to the filament-to-filament interface. SEM (Scanning Electron Microscope) images Fig. 6.14c corroborate the existence of extended cracks between filaments in this ABS object, a Stanford Bunny.



(a) (b) (c)

Figure 5.5: (a)nose-down vertical absorption 3D bunny volume, (b)nose-down vertical dark-field shell-masked 3D bunny volume with filaments and (c) SEM of FDM (Fused deposition modeling) filaments from a portion of the ear of the Stanford Bunny showing cracks between the ABS filaments (see insert).

5.3 Dragonfly Workflow

5.3.1 Introduction

Neutron interferometry/tomography is being developed for non-destructive evaluation of additive manufacturing samples undergoing stress and fatigue testing, an application requiring an efficient workflow. A well-defined workflow is needed for several reasons: The instrument time is precious, with only a few beamlines available world-wide. The data volume ranges from moderate to large. The data and image processing has several complex steps: First, the reduction of the interferograms to projections, possibly by a vectorized least squares algorithm [18]. Second, the inverse Radon transforms for the three imaging modalities often use more than one method, e.g., a Fourier method for absorption and an iterative method for the dark-field (scattering) projections [19]. Third, a volume registration with the coordinate system of the sample is required so as to correlate tomography with other physical measurements such as SEM and optical microscopy [20]. Fourth, the dark-field (scattering) volume, when normalized with the attenuation volume, provides information about flaws within the additive manufacturing samples, flaws such as porosity and crack formation [21, 22, 23, 24].

In a recent project [18], our workflow was a sequence consisting of Mathematica notebook, a TomoPy/ASTRA/Jupyter notebook, and a second Mathematica notebook. This workflow suffices for research purposes, but not appropriate for production work. In particular, the coordinate system registration was difficult and the potential for error is significant.

Herein, the definition of "workflow" is in the sense developed in a recent review of over 260 scientific workflows. In Garijo's paper [1], the authors have manually analyzed over 260 workflows encoded in open source and commercial software to extract common features. The results show two major motifs, data input/output and calculations, which are described by a motif ontology subdivided into categories such as "data cleaning", "atomic workflows", and "human interaction". Data cleaning and human interaction are well known by the tomographer; atomic workflow refers to a hands-off computation such as submitting

a list of cleaned and centered sinograms for slice reconstruction. Therefore, we use the motif ontology, including color coding of flowcharts, of Garijo et al. to describe workflows and apply the workflow template to our dogbone datasets.

5.3.2 Workflow

Interferograms must be processed to convert recorded images into projections; then used TomoPy/ASTRA to reconstruct into usable volumes. Future users should take steps interferograms→TomoPy/ASTRA→Dragonfly.

Fig. 6.11 describes the portion of workflow implemented in Dragonfly from reconstructed volumes in TomoPy/ASTRA to quantitative analysis. Four main steps, import, alignment(optional), binarization and arithmetic are conducted in this workflow. Our input sources include reconstructed absorption(attenuation) and dark-field volumes, obtained from TomoPy/ASTRA/Jupyter notebook output. Our goal is to examine internal features (e.g. fractures, pores, cracks). Significantly, this scientific workflow could be extended to any interferometry/tomography imaging analysis. In the beginning of the workflow in Fig. 6.11, Row 1 starts with attenuation and dark-field volumes computed in TomoPy/ASTRA.

The action from row 1 to row 2 is to identically crop the volumes where the region of interest (ROI) was defined manually. In Dragonfly, one opens the "Dataset Cropper panel" and crops ROI based on the bounding box defined with the clip box tool. Shown in Fig. 6.11 are representative thumbnails, typically of the fractured end of the shortest sample visible in Fig. 4.4.

Dragonfly has efficient tools for image and volume alignment. In this particular experiment, alignment is unnecessary and is shown as optional between row 2 to row 3. The alignment toolbox in Dragonfly is robust to align the region of interest along the Z-axis of the current selected view via rotation and translation. The alignment tools were used when processing and merging two other tomography volumes acquired with an orthogonal sample orientation [18]. After alignment, the two volumes are registered each other and

ready for binarization.

Row 3 to row 4 to row 5 spans the binarization of the absorption volume to create a binary mask. The diamond notation of Garjio for binarization indicates an interaction with user-adjustable parameters. The mask can, with erosion, focus attention on internal cracks or, with the difference between two mask, focus attention on surface cracks. The mask are generated from the absorption volume due to the greater image contrast to noise ratio versus the dark-field image. The crack analysis uses masked, but unfiltered dark-field volume (row 5); a corresponding masked, filtered dark-field volume is used for display. "ROI Painter Tools" in Dragonfly offer dilation, erosion, and opening/closing operations to create a clean masked module.

The masked absorption volume at row 5 is used to define the laboratory coordinate system. In particular, line probes along the short and long fractured samples are used to relate volume slice numbers to a laboratory axes system with units of millimeters (see Fig. 5.7).

Row 5 to Row 6 shows an arithmetic macro created upon user's request. In this paper, our arithmetic macro is to compute a division (1-DF)/A, where DF is dark-field volume and A is absorption volume. This macro was implemented in the imaging toolbox control system in Dragonfly. Since dark-field (scattering) is more sensitive to beamlines than absorption, this arithmetic macro would denote a certain relationship between dark-field volume and absorption volume. Meanwhile, (1-DF)/A volume indicates strong intensity signals so as to detect potential crack information in samples.

Row 6 to Row 7 displays a powerful macro for sample quantitative analysis, Slice Analysis with ROI tools, to help explore the potential fractures of the internal stressed sample Fig. 5.8. ROI tools, selected sample of rasters, are useful for region of interest analysis. In our sample, the stressed bones was painted with ROI tools for slice analysis. Slice analysis macro includes mean, median, max/min, standard deviation etc. arithmetic calculation methods for each slice in volumes. Afterwards, a line profile will be created

to connect those point in each slice. Herein, slice analysis is a fast and efficient macro for quantitative image analysis.

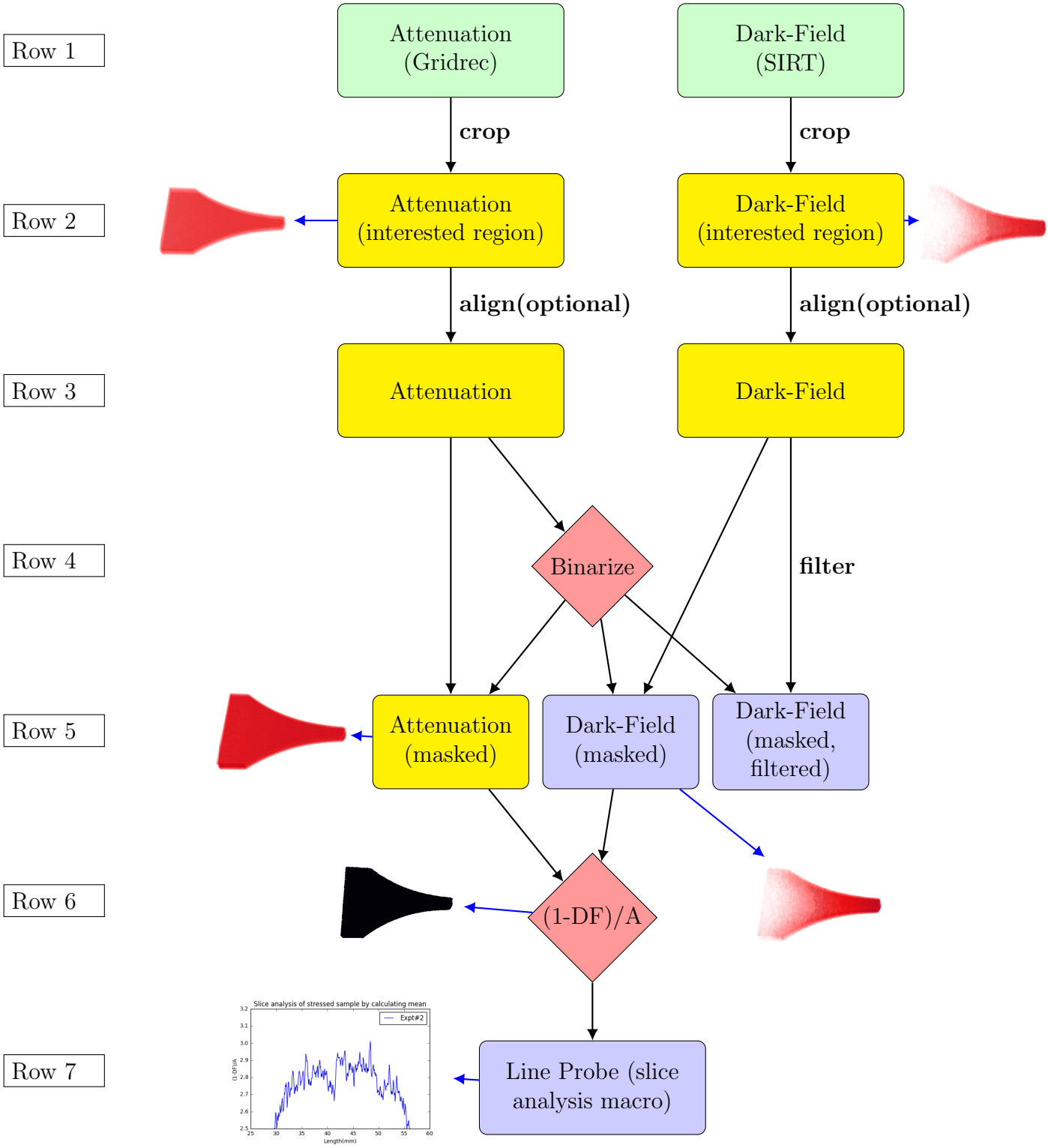


Figure 5.6: Second portion of workflow for processing reconstructed volumes with Dragonfly Based on the scientific workflow motifs in Garijo et al. paper [1], arrows represent manipulations, green rectangles represent input augmentation; yellow rectangles represent data analysis, red diamonds represent parameter adjustment and blue rectangles represent output extraction.

5.3.3 Macros

Towards the goal of robust workflow development, Dragonfly includes a macro engine. Repetitive image handling tasks can be stored and efficiently executed on demand through use of Dragonfly's macro tools. A macro, a sequence of operations, is easily recorded with the macro recorder than runs while users go about their standard interactive use of the software. Each sequential elementary operation (e.g. loading an image from disk, thresholding an image, executing an image filter, etc) has a corresponding Python command that is logged in a named macro. This works to the strong benefit of standardized workflow development. Since advanced users with more Dragonfly expertise understand the required operations, they can perform them once, and then the procedure is encoded in a macro that less experienced users can reproducibly execute to produce the same results.

The workflow outlined in Fig. 6.11 was easily captured with the macro recorder where each arrow depicts an elementary operation that is defined as a macro step. Each of these operations takes one or more images as input and returns one or more images as output. The Python source code for the macro reflects these inputs as arguments to the invoked function calls, and the outputs are reflected as return values. The macro engine is programmed with a continuity self-awareness that understands when the output of one operation is expected to be used as input to a subsequent operation. The non-image function arguments may include other parameters such as image filter kernel size, or the filename of a file to be loaded or saved, etc. All of the function parameters can be interactively inspected and tuned prior to executing the macro because they are exposed to the user as part of the macro player interface.

In order to maximize workflow flexibility, users can execute the macro from beginning to end without interruption, or they can proceed stepwise. Additionally, the macro player permits users to add or remove breakpoints, where execution of the macro will pause for the operator. This means that when playback is paused, the user can inspect the progress thus far so he or she can make informed decisions about whether to alter downstream parameters,

or even interactively perform other operations in the software, before resuming playback of the macro.

Python is not a statically typed language so–without proper software engineering in the macro engine–there exists a risk that during playback users might replace an integer parameter with a string parameter and the ensuing behavior would be ill-defined. This is addressed in Dragonfly by ensuring that every operation that is logged also includes documentation strings that specify all of the parameters and their types. This can be seen in Table. 5.1, which shows what is recorded for a single macro step. Not only is the specific function–in this case, the binarize operation is expressed as the *ROITools.addRange()* function (line 31)–defined, but all of the input parameters and their types are documented as well. Consequently, the macro player knows precisely what variables are permissible when users wish to adjust parameters during playback. When it comes to customizing the macro behavior, more important than the ability to change individual parameters, is the fact that the entire macro is in pure Python. This means Python-savvy users can take full advantage of the Python programming language to customize the macro. Users might manually insert other steps which invoke external library functions; they may define local functions that get invoked throughout the macro; they may insert logging or debugging code, etc.

It is noteworthy that Dragonfly's macro recording is not restricted to operations that generate an output. When users change the on-screen display by toggling the visibility of an image, zooming or panning, or even changing the brightness and contrast of an image, all of those changes are captured and encoded in the macro. If the macro author does not wish to preserve them as part of the defined workflow, he or she can simply click the Delete button found next to each elementary operation in the macro player interface. If the macro user does not wish to execute certain steps, he or she can click the Skip button found next to each macro step so that macro execution will simply skip to the subsequent step when that part of the workflow is reached.

```
********** BEGIN MACRO ********** #
"""
Adds each voxel where the intensity in the dataset
is included in the given range.

:name: addRange
:execution: execute

:param listROIToModify_1: ROI to modify
:type listROIToModify_1: ORSModel.ors.ROI
:count listROIToModify_1: [1, None]
:param derivedDataset: Dataset for range comparison
:type derivedDataset: ORSModel.ors.Channel
:param rangeMin: minimal value of the range
:type rangeMin: float
:param rangeMax: maximal value of the range
:type rangeMax: float
:param timeStep_1: timeIndex
:type timeStep_1: int
"""

# ----- BEGIN INPUT ARGUMENT DEFINITION ----- #
listROIToModify_1 = [output_3]
rangeMin_1 = 10.0
rangeMax_1 = 20.0
timeStep_1 = 0

# ----- END INPUT ARGUMENT DEFINITION ----- #
# Interface method
OrsVolumeROITools.addRange(listROIToModify=listROIToModify_1,
                           dataset=derivedDataset,
                           rangeMin=rangeMin_1,
                           rangeMax=rangeMax_1,
                           timeStep=timeStep_1)

OrsVolumeROITools.addRange( [output_3], derivedDataset, 10.0, 20.0, 0)

# ********** END MACRO ********** #
```

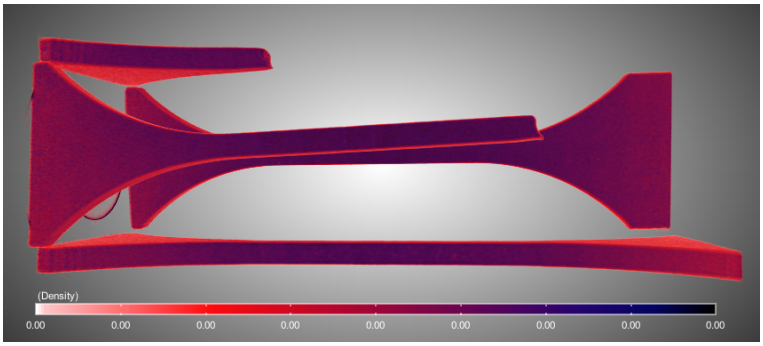
Table 5.1: A macro for binarization analogous to the routine used in Row 4, Fig. 6.11.

5.3.4 Results and Discussion

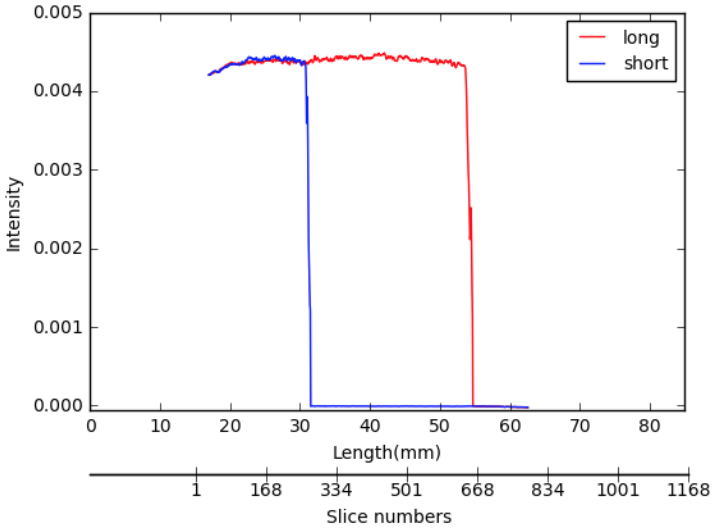
In order to characterize the results of the tension experiments, we wish to rely on both visual and quantitative interpretations of the reconstructed neutron tomography image data. The volume rendering of the attenuation reconstructions shows the transition of intensity from sample to air, while dark-field reconstruction indicates local areas of increased intensity (colored in green) that highlight the effect of the tension.

Profiling the intensity along a straight line is a standard method for evaluating images. This kind of line probe technique is used to understand how the image intensity changes as the probe crosses over material boundaries. As shown in Fig. 5.7(b), the line probe follows the absorption signal along the dogbone axis and shows the rapid intensity falloff as the probe crosses from sample to air, where the attenuation intensities of air fluctuate near zero. By visual measurement of the dogbone samples in the laboratory, we know the lengths of short and long fragment are 31.4mm and 54.6mm, respectively. In Fig. 5.7(b), the gap for short sample is found at a length of around 31mm, nearly matching the measured 31.4mm; while for the longer, the gap appears at 54mm, again close to the laboratory measurement, 54.6mm. Particularly, our interferometry data was collected far away approximately 15 mm from the base of the sample in Fig. 5.9(b).

The shortcoming of simple line probes is that because they sample only directly along the path of the linear probe, their limited sampling is prone to noise. A statistically superior approach is to sample many pixels in the neighborhood of the probe. This task maps well on to the Dragonfly tool for Slice Analysis. Each slice, sampled coaxially to the line probe is computationally evaluated with arbitrary measurements. In this case, the mean intensity of every dogbone pixel from that slice is computed and every air pixel from that slice is ignored. By aggregating the signal across the entire observed slice of the dogbone, we benefit from greater signal-to-noise as seen in Fig. 5.9(a).



(a)



(b)

Figure 5.7: LineProbe through the tips of the short (31.4 mm) and long (54.6 mm) portions of the vertical-grating sample are used as fiducial points for the X-ray distance scale (a) The attenuation volume shows most of the 80 mm long dogbones; the base of the sample, Fig. 5.8, is to the left. (b) The tip ends of the short and long samples establish the correspondence between slice number and a distance scale in millimeters.

Dark-field volume denotes evident scatterings due to the fractures of sample. From the Fig. 5.8(a), the short end and long end indicate strong scatterings in the 3D dark-field volume. In the Fig. 5.8(b), with slice analysis macro in Dragonfly, we quantitatively calculate mean value for each slice in volumes and create a line profile for the stressed dogbone.

Meanwhile, the scatterings (green) shown in the stressed sample characterize early crack formation inside the stressed sample, which is the region of interest in slice analysis in Fig. 5.9. Especially, in Dragonfly, we manually labeled the early crack point in 2D slice and the label would be shown in 3D view.

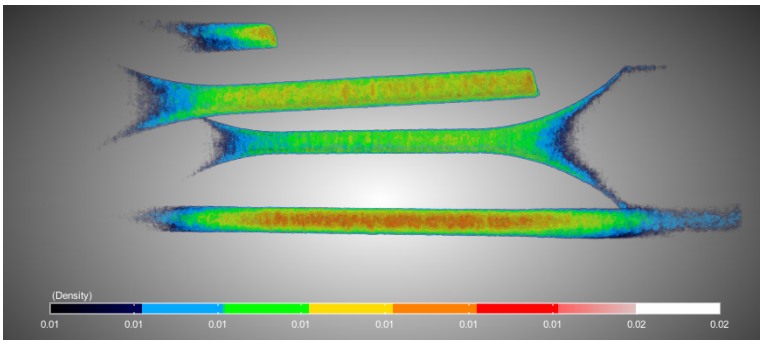
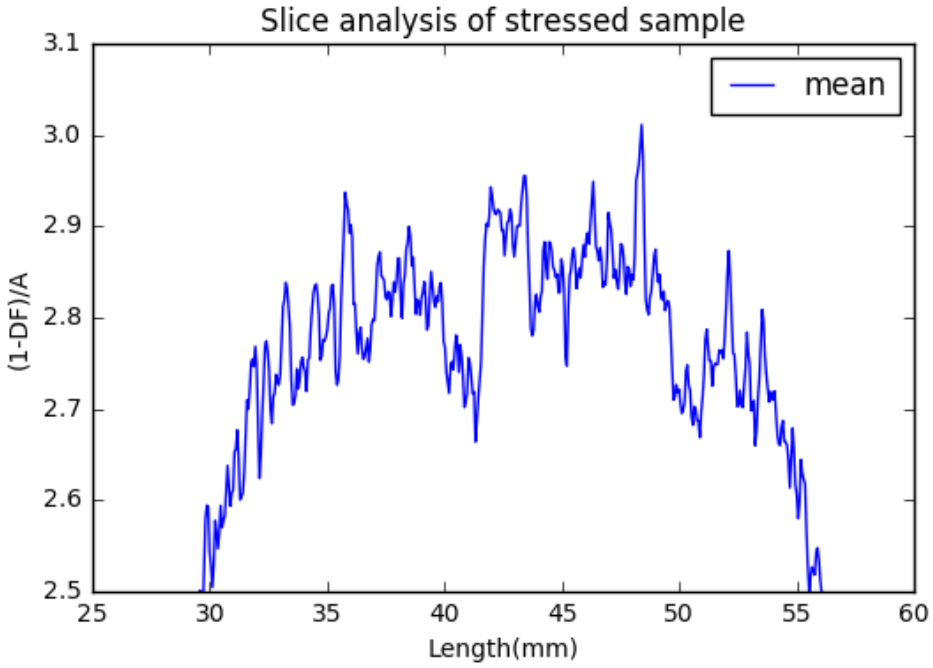
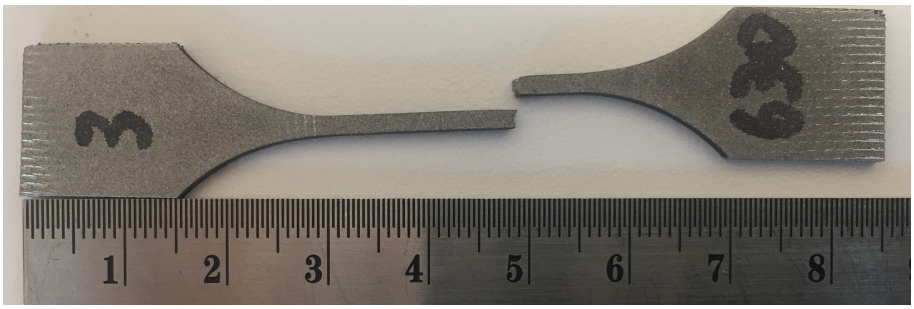


Figure 5.8: 3D dark-field volume rendering of vertical-grating samples in Dragonfly: short, long, pristine and stressed dogbones.

To explore more about the fractures in the stressed sample, by using slice analysis macro, we made a line-probe of the stressed sample between the end of short sample and the end of long sample for different vertical-grating. From the line probe, we figured out the spike approximately located at the length of 48mm in Fig. 5.9 (a), where the potential fractures lie in, approximately corresponding to the real position 48.75mm of the stressed sample fracture after additional tensile stress to failure in Fig. 5.9(b).



(a)



(b)

Figure 5.9: (a) Line probes of slice analysis from vertical-grating. The line probes go through short sample end to the long sample end. The volume (1-DF)/A has been masked to exclude air values. (b) The stressed sample after additional tensile stress to failure. The sample lengths are 48.74mm (left piece) and 36.64mm (85mm assembled) [18].

The neutron imaging experiment can observe only a small volume. The resulting reconstruction, therefore, has limited spatial extent. In our case, we have a field-of-view that does not span the full length of the dogbone samples. Despite the limited field-of-view, we should be able reconcile the observed reconstruction in a proper frame of reference that matches the laboratory frame.

In order to interchangeably relate the laboratory measurements with the measurements in the reconstructed images, i.e. digital domain, we would be best served if we had the digital domain coordinate system match the laboratory frame of reference. We define that laboratory frame here with the rotation axis of the tomography experiment as the laboratory Z-axis, with the Z-origin found at the base of the mounted samples. For our purposes, we have no need to constrain the x and y axes.

Objects and images manipulated in Dragonfly have spatial coordinates in the Dragonfly coordinate system. In order to have the Dragonfly digital domain accurately mirror the laboratory frame of reference, we must ensure that our reconstructed volume is loaded into Dragonfly with the proper orientation and the proper origin, and that the image voxel size is accurate. If these three constraints are satisfied, then we can make measurements in either domain, and they should be consistent across both.

Most users input the voxel size when importing images into Dragonfly so that constraint is usually satisfied by routine image import. In the following text, we describe how to satisfy the constraints of orientation and origin.

It was out of convenience that we chose the rotation axis of our tomography experiment to be the laboratory Z-axis. The reconstruction algorithm we relied on creates a digital 3D volume, which, by default, uses the sample rotation axis as the reconstructed digital Z-axis. When we imported the reconstruction into Dragonfly, we were guaranteed that our imported image would be axially aligned (either parallel or anti-parallel) to our digital frame. For imported images that have no encoded origin, the first plane of reconstructed pixels is taken as Z=0, which locates the bottom of our imported image at the spatial origin

for our digital domain.

Upon interactive investigation, one can easily determine if the axial alignment is consistent with the laboratory frame. In our case, we made two digital measurements that should have been ascending, the Z-position of the tip of the short dogbone fragment and the Z-position of the tip of the long dogbone fragment. Since we observed them to have descending digital Z coordinates, we applied a Z-axis inversion that immediately reconciled the orientation with the laboratory frame.

Because our reconstructed field-of-view does not include the bottom 15mm of the sample, the first plane of pizels digital Z-coordinate of 0mm corresponded to Z=15mm in the laboratory frame. We corrected for this in Dragonfly by manually inputing the Z-origin for our reconstructed volume; we simply entered a value of 15mm in the Dataset Properties Advanced dialog which exposes the x,y,z origin coordinates for any image channel for user editing.

With the orientation and origin correct, we were then able to make accurate measurements. We found the tip of the short dogbone fragment to be at Z=31.4 and the tip of the long dogbone fragment to be at Z=54.6.

5.3.5 Conclusions

The interferometry/tomography workflow for additive manufacturing testing is developing into a two-step process. First, a GUI-driven visualization tool is used to develop the analysis. Then, a command line macro is used to repeatedly perform the analysis over multiple samples. The balance of GUI and macro interactions is needed for flexibility, speed, and (MPM: I don't think we can make claims about data provenance) provenance of actions.

In the paper, we have demonstrated that Dragonfly successfully and effectively incorporated a friendly GUI and Python-based macros for interferometry/tomography image analysis. Dragonfly offers 3D view, binarization, ROI tools, line profiles and macros, allowing full flexibility in the workflow. Particularly, macros for Python developers facilitate

image processing, which can be recorded step by step in Dragonfly for future use.

Our Dragonfly results demonstrate the fractures (early crack information) do exist in the stressed sample, air-sample transition in the absorption volume as well as scatterings in the dark-field volume. Meanwhile, macro engine takes detailed procedures and makes them easy, reproducible, robust, without sacrificing flexibility, roles for experts and novices. Visualization and Quantitative Analysis, such as Slice Analysis, has quantitatively displayed the internal structure in dark-field volumes and potential cracks in the stressed sample. Moreover, Dragonfly has successfully reconciled the observed reconstruction digital frame to the laboratory frame.

At present, ORS company only released Windows version for Dragonfly software. Linux and MacOS would be available soon in the next couple of years. Scientists from NIST (National Institute of Standards and Technology in U.S.) consider to incorporate our Jupyter notebook for image reconstruction and Dragonfly for 3D visualization into their interferometry/tomography system. Nevertheless, a couple of disadvantages in Dragonfly, quality of binarization and instability due to overtime of graphical card, need to be improved in the future. In general, Dragonfly is a reliable software for interferometry/tomography 3D visualization in the world.

5.4 References

[1] Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., Goble, C. (2014) Common motifs in scientific workflows: An empirical analysis. Future Generation Computer Systems-the International Journal of Grid Computing and Escience **36**: 338–351.

[2] Navarro, A.K.W., Vassiliadis, V.S. (2014) "computer algebra systems coming of age: Dynamic simulation and optimization of dae systems in mathematica (tm)". Computers & Chemical Engineering **62**: 125–138.

[3] Tauxe, L., Shaar, R., Jonestrask, L., Swanson-Hysell, N.L., Minnett, R., Koppers, A.A.P., Constable, C.G., Jarboe, N., Gaastra, K., Fairchild, L. (2016) Pmagpy: Software package for paleomagnetic data analysis and a bridge to the magnetics information consortium (magic) database. Geochemistry Geophysics Geosystems **17**(6): 2450–2463.

[4] Grsoy, D., De Carlo, F., Xiao, X., Jacobsen, C. (2014) Tomopy: a framework for the

analysis of synchrotron tomographic data. Journal of Synchrotron Radiation **21**(Pt 5): 1188–1193.

[5] van Aarle, W., Palenstijn, W.J., De Beenhouwer, J., Altantzis, T., Bals, S., Batenburg, K.J., Sijbers, J. (2015) The astra toolbox: A platform for advanced algorithm development in electron tomography. Ultramicroscopy **157**: 35–47.

[6] van Aarle, W., Palenstijn, W.J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Joost Batenburg, K., Sijbers, J. (2016) Fast and flexible x-ray tomography using the astra toolbox. Optics Express **24**(22): 25129–25147.

[7] Pelt, D.M., Grsoy, D., Palenstijn, W.J., Sijbers, J., De Carlo, F., Batenburg, K.J. (2016) Integration of tomopy and the astra toolbox for advanced processing and reconstruction of tomographic synchrotron data. Journal of Synchrotron Radiation **23**(Pt 3): 842–849.

[8] Klukowska, J., Davidi, R., Herman, G.T. (2013) Snark09 a software package for reconstruction of 2d images from 1d projections. Computer Methods and Programs in Biomedicine **110**(3): 424 – 440.

[9] Kaestner, A.P. (2011) "muhrec–a new tomography reconstructor". Nucl. Instrum. Methods A **651**(1): 156–160.

[10] Klukowska, J., Davidi, R., Herman, G.T. (2013) "snark09 and 2013; a software package for reconstruction of 2d images from 1d projections". Computer Methods and Programs in Biomedicine **110**(3): 424–440.

[11] Mader, K.S., Schneider, P., Muller, R., Stampanoni, M. (2013) A quantitative framework for the 3d characterization of the osteocyte lacunar system. Bone **57**(1): 142–154.

[12] Mader, K., Stampanoni, M. (2016) "moving image analysis to the cloud: A case study with a genome-scale tomographic study". AIP Conference Proceedings **1696**(1): 020045.

[13] Morisette, J.T., Jarnevich, C.S., Holcombe, T.R., Talbert, C.B., Ignizio, D., Talbert, M.K., Silva, C., Koop, D., Swanson, A., Young, N.E. (2013) "vistrails sahm : visualization and workflow management for species habitat modeling". Ecography **36**(2): 129–135.

[14] West, A.M., Kumar, S., Jarnevich, C.S. (2016) "regional modeling of large wildfires under current and potential future climates in colorado and wyoming, usa". Climatic Change **134**(4): 565–577.

[15] Tohline, J.E., Santos, E. (2010) "visualizing a journal that serves the computational sciences community". Computing in Science & Engineering **12**(3): 78–81.

[16] Starck, J.L., Elad, M., Donoho, D.L. (2005) "image decomposition via the combination of sparse representations and a variational approach". IEEE Transactions on Image Processing **14**(10): 1570–1582.

[17] Bertalmio, M., Vese, L., Sapiro, G., Osher, S. (2003) "simultaneous structure and texture image inpainting". IEEE Transactions on Image Processing **12**(8): 882–889.

[18] Brooks, J.A., Knapp, L.G., Yuan, J., Lowery, G.C., Pan, M., Cadigan, E.B., Guo, S., Hussey, S.D., Butler, G.L. (2017) Neutron imaging of laser melted ss316 test objects with spatially resolved small angle neutron scattering. Journal of Imaging **3**(4).

[19] Ritter, A., Anton, G., Weber, T. (2016) Simultaneous maximum-likelihood reconstruction of absorption coefficient, refractive index and dark-field scattering coefficient in x-ray talbot-lau tomography. PLoS ONE **11**(10): e0163016. PONE-D-15-20532[PII] 27695126[pmid] PLoS One.

[20] Lucas, M.S., Gnthert, M., Gasser, P., Lucas, F., Wepf, R. In: Chapter 17 - Bridging Microscopes: 3D Correlative Light and Scanning Electron Microscopy of Complex Biological Structures. Volume 111. Academic Press (2012) 325–356

[21] Slotwinski, J.A., Garboczi, E.J., Hebenstreit, K.M. (2014) Porosity measurements and analysis for metal additive manufacturing process control. Journal of Research of the National Institute of Standards and Technology **119**: 494–528. jres.119.019[PII] 26601041[pmid] J Res Natl Inst Stand Technol.

[22] Eckel, Z.C., Zhou, C., Martin, J.H., Jacobsen, A.J., Carter, W.B., Schaedler, T.A. (2016) Additive manufacturing of polymer-derived ceramics. Science **351**(6268): 58.

[23] Terriault, P., Brailovski, V. (2018) Modeling and simulation of large, conformal, porosity-graded and lightweight lattice structures made by additive manufacturing. Finite Elements in Analysis and Design **138**: 1–11.

[24] Brnler, R., Aibibu, D., Wltje, M., Anthofer, A.M., Cherif, C. (2017) In silico modeling of structural and porosity properties of additive manufactured implants for regenerative medicine. Materials Science and Engineering: C **76**: 810–817.

Chapter 6
2D Phase Integration

6.1 Introduction

X-ray interferometry has been rapidly developed within the past a few years. The notable advantage of X-ray interferometry is to produce conventional attenuation images as well as another two imaging modalities, differential phase contrast (DPC) and dark field (DF). As for DPC, it shows stronger sensitivity to low atomic number elements, more valuable to structure analysis than absorption. However, the quality of DPC images usually denotes poor contrast due to noisy background and unstable gratings. As a result, no much useful information can be gained from DPC images. In this case, a new technique, X-ray phase imaging is prominent enabling the observation of structures in materials. As for our microstructure sample, X-ray phase images were obtained with Harker-O'Leary algorithm, a 2D differential phase integration method for vertical grating and horizontal grating images.

In Atsushi Momose's paper [1], a mathematical algorithm for phase shift $\Phi(x, y)$ was proposed as

$$\phi(x, y) = \frac{2\pi}{\lambda} \int \delta(x, y, z) dz \tag{6.1}$$

where λ is X-ray wavelength, $\delta(x, y, z)$ is refractive index decrement from internal structure that X-ray goes through. In view of atomic points, $\delta(x, y, z)$ can be expressed as

$$\delta(x, y, z) = \frac{r_e \lambda^2}{2\pi} \sum_k N_k(x, y, z)(Z_k + f'_k) \tag{6.2}$$

If combine equation (1) and (2), we can get

$$\phi(x,y) = \int \sum_k N_k(x,y,z)\rho_k dz \tag{6.3}$$

where $\rho_k = r_e\lambda(Z_k + f_k')$. Now, the cross section of X-ray phase shift $\phi(x,y)$ can be attributed to parameter ρ_k. To explore the sensitivity of ρ_k varying with atomic number, Atsushi plotted interaction cross sections of phase shift ρ vs the atomic number. Compared with cross section of absorption, phase imaging obtains an extremely high sensitivity as a result of higher interaction cross section.

In this case, phase shift is attractive for object investigation, we consider applying phase shift algorithm to our sample, foraminifera, a microstructure. Since the differential phase contrast (DPC) images for vertical and horizontal gratings are influenced by noisy background due to unstable instrument, it is our first time to attempt Harker and O'Leary algorithm in order to get clean phase images. Harker-O'Leary algorithm adopted Tikhonov regularization which was demonstrated as a Sylvester equation [2, 3]. Matlab package provided by Harker and O'Leary [4, 5] has been used for DPC 2D integration. Other traditional 2D integration methods, Frankot-Chellappa algorithm [6] and Poisson solver [7] are special cases for surface reconstruction solution; meanwhile, noise and phase wrapping are hard to be eliminated. In comparison to those traditional methods for 2D phase integration, Harker-O'Leary with Tikhonov regularization is more robust.

6.2 Theory of the Harker-O'Leary algorithm

Tikhonov regularization [3, 8, 9] is to solve the problems in 2D matrix domain, matrix-based discretization of regularization terms for 2D systems. The functions tends to find the minimum value ϵ of functional, where λ is a positive constant.

$$\epsilon(y) = ||Ay - b||_2^2 + \lambda||S_y||_2^2 \tag{6.4}$$

$$(A^T A + \lambda S^T S)y = A^T b \tag{6.5}$$

The equation (2) is derivative from equation (1) upon rearranging yields. We demonstrated the Tikhonov regularization could be written by Sylvester Equation. In our Matlab scripts [1], we've generated the differentiation matrices and solve the Sylvester Equation with "lyap" in Matlab control systems toolbox. Here is a sample function *g2sTikhonov_simple* to generate a regularization matrix.

```
% Generate the Differentiation Matrices and Solve the Sylvester Equation
    tol = sqrt( eps(1) ) ;
    A = [ Dy ; mu * speye(length(y)) ] ;
    B = [ Dx ; lam * speye(length(x)) ] ;
    F = [ Zy ; mu * Z0 ] ; % Degree-0 means Dx^0 = I
    G = [ Zx , lam * Z0 ] ;

% lyap is in the Matlab Control Systems Toolbox
    Z = lyap( full(A'*A), full(B'*B), -A'*F - G*B ) ;

% Compute the Residuals:
    Res = zeros(2,2) ;
    Res(1,1) = norm( Z * Dx' - Zx, 'fro' ) ;
    Res(1,2) = norm( Dy * Z  - Zy, 'fro' ) ;
if strFileNamePrefix =='logo_micro_',
    disp(['A (converted from sparse to full),  ',num2str(size(A))]);
    disp(full(A));
    disp(['B (converted from sparse to full),  ',num2str(size(B))]);
    disp(full(B));
```

[1] https://github.com/jyuan4/2D_DPC_Integration

```
    disp(['F ,  ',num2str(size(F))]);
    disp(F);
    disp(['G ,  ',num2str(size(G))]);
    disp(G);
    disp(['Z ,  ',num2str(size(Z))]);
    disp(Z);
end;
```

The Frobenius norm of the solution for function $z = z(x, y)$ is the most common regularization term, where z is the Laplacian.

$$\rho(Z) = ||Z||_F^2 = \frac{1}{2}(||I_m Z||_F^2 + ||Z I_n||_F^2) \tag{6.6}$$

after directional derivative of this function, its discretization is,

$$\rho_1(Z) = ||D_y Z||_F^2 + ||Z D_x^T||_F^2 \tag{6.7}$$

with multiple-steps matrix calculations, finally, the regularization term will be,

$$\rho_2(Z) = ||D_y^2||_F^2 + ||Z(D_x^2)||_F^2 \tag{6.8}$$

The least square surface 2D system reconstruction with Tikhonov Regularization was proved to be a Sylvester equation, which was used to solve the ρ in equation (5). Therefore, the functional for Tikhonov Regularization is,

$$\epsilon(Z) = ||D_y Z - \hat{Z}_y||_F^2 + ||Z D_x^T - \hat{Z}_x||_F^2 + \lambda(||S_y Z||_F^2 + ||Z S_x^T||_F^2) \tag{6.9}$$

where Z is the cost function and after differentiation of Z, the equation above yields the

corresponding normal equations, which is known as a Sylvester Equation,

$$(D_y^T D_y + \lambda S_y^T S_y)Z + Z(D_x^T D_x + \lambda S_x^T S_x) - D_y^T \hat{Z}_y - \hat{Z}_x D_x = 0 \tag{6.10}$$

6.3 A Naive Model

To proof the correction of MATLAB scripts of Harker-O'Leary algorithm acquired from APS (Advanced Photon Source) institute, we built a naive model in the beginning. First, we draw a 3D Matlab logo and name it as a function $f(x, y, z)$; then differentiate $Z_x N = \frac{d_z}{d_x}$ and $Z_y N = \frac{d_z}{d_y}$ respectively.

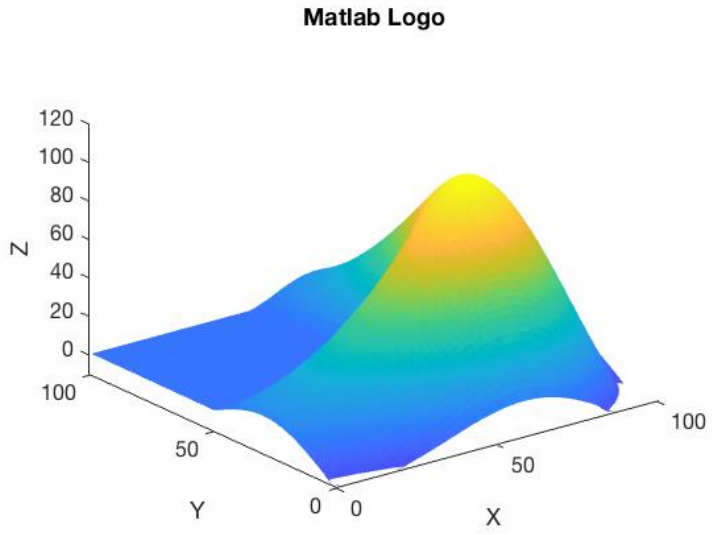


Figure 6.1: Matlab logo

Second, we add noise to $Z_x N$ and $Z_y N$ considering the real situation.

$$Z_x N = Z_x + \sigma \times A_x \times randn(m, n) \tag{6.11}$$

$$Z_y N = Z_y + \sigma \times A_y \times randn(m, n) \tag{6.12}$$

where

$$A_x = \frac{(max(Z_x(:)) - min(Z_x(:)))}{2}$$
$$A_y = \frac{(max(Z_y(:)) - min(Z_y(:)))}{2}$$
$$m = min(size(Z_x))$$
$$n = m - 1$$

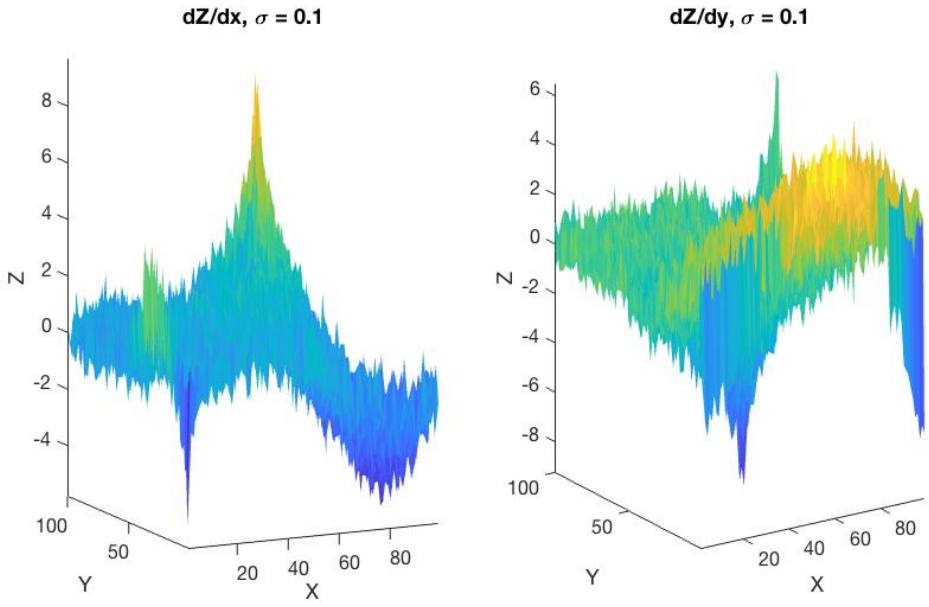
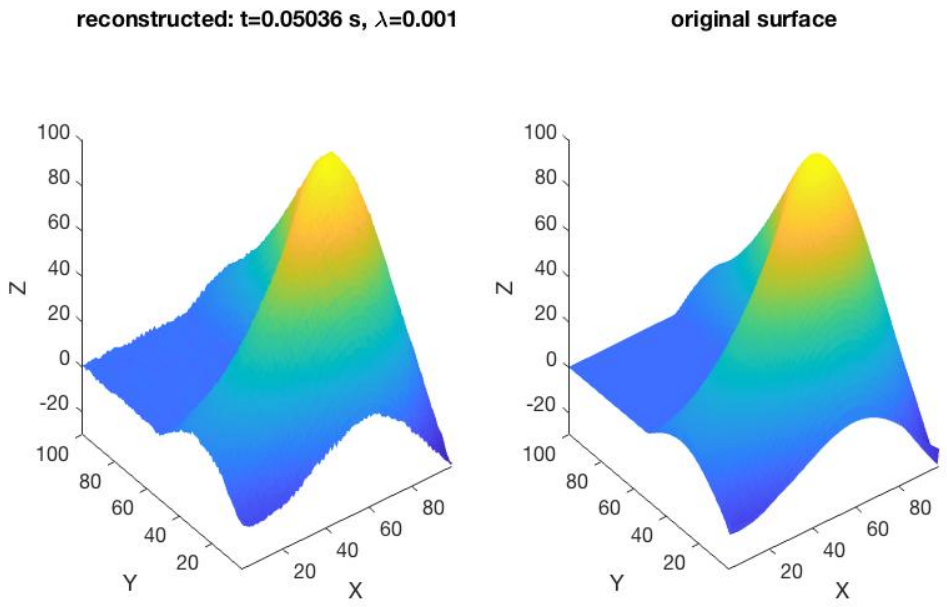
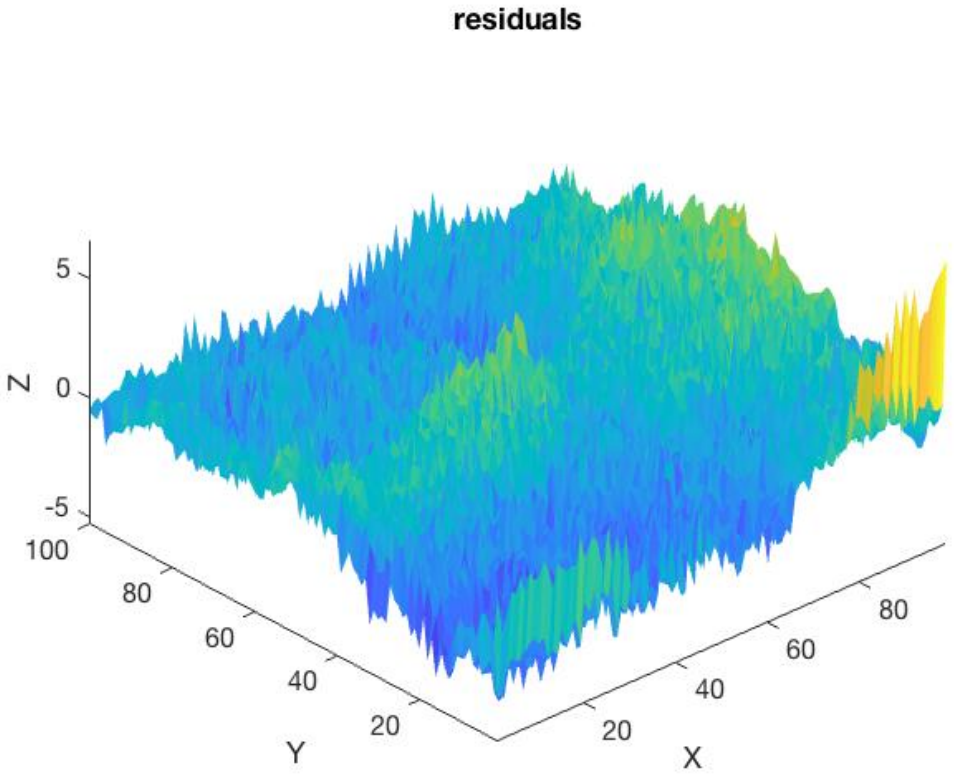


Figure 6.2: Differentiation of Matlab logo in X-axis and Y-axis

Third, with Tikhonov Regularization to simply computes the global least squares reconstruction of a surface, we obtain the integrated Matlab logo from noisy differentiation function Z_xN and Z_yN. If we calculate the residual between integrated Matlab logo and the real logo, we could find the difference is extremely small. In the case of small residual, the Harker-O'Leary Matlab script is correct and ready to be utilized into our experimental datasets.



(a)



(b)

Figure 6.3: (a) Comparison of integrated and original Matlab logo (b) residual of the two Matlab logos

6.4 A Simulation Model

We have simulated a synthetic sphere DPC data for both horizontal and vertical gratings. Next, we project the 3D volume into a 2D image and calculated DPC without phase

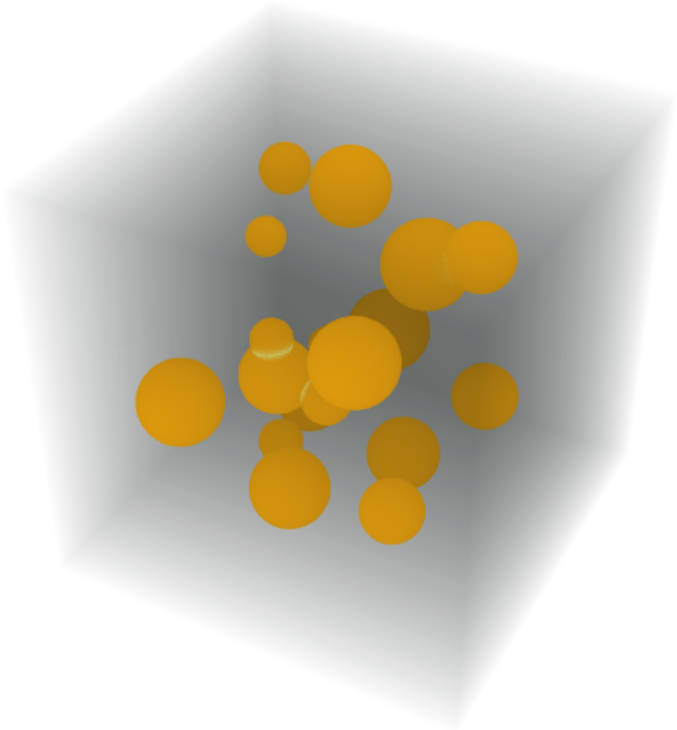


Figure 6.4: A 3D synthetic volume of phase objects: 21 spheres.

wrapping in both horizontal and vertical directions.

After 2D integration with Harker-O'Leary algorithm in Matlab, we generated phase projection images.

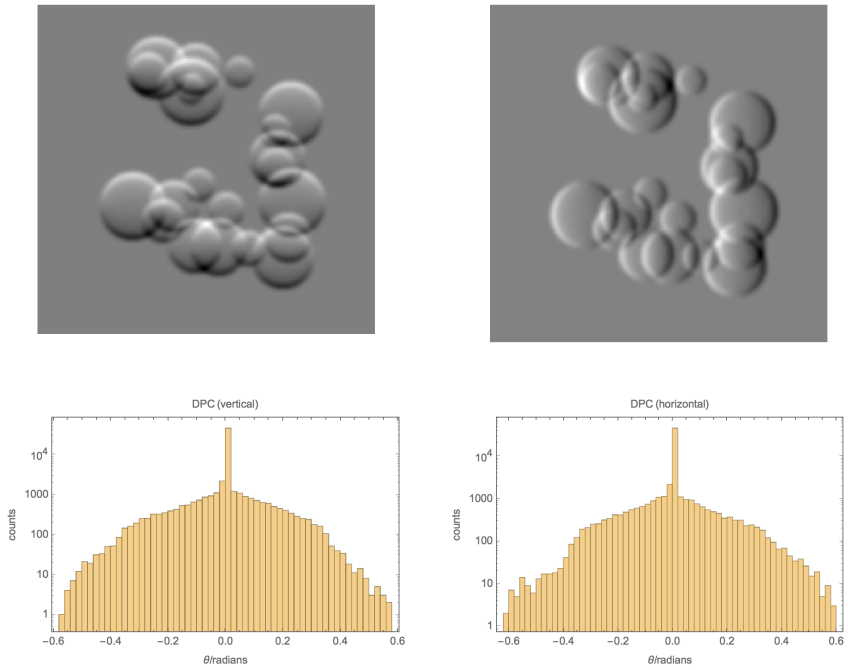
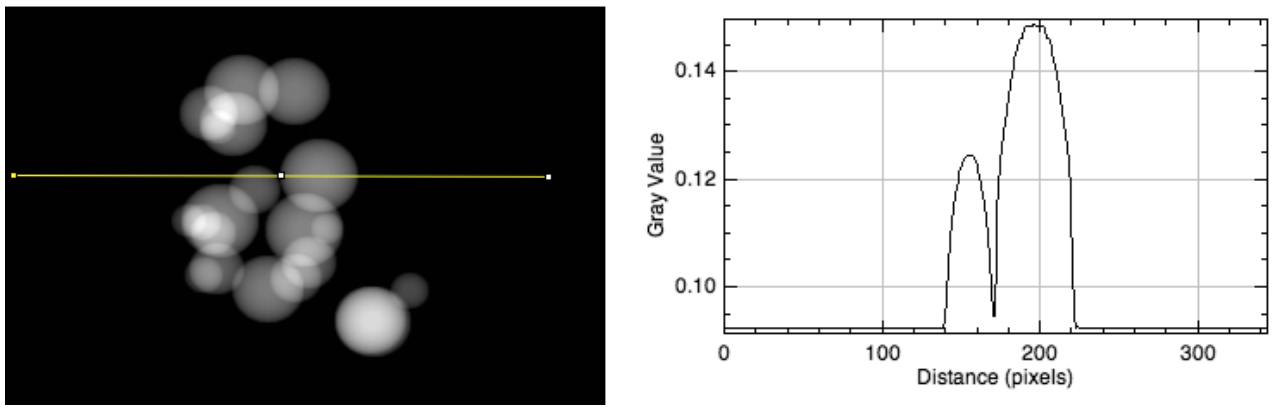


Figure 6.5: Unwrapped DPC images and histograms of horizontal and vertical gratings



(a) (b)

Figure 6.6: Synthetic phase projection image at 0^o (left) and corresponded line probe (right)

6.4.1 Effect of Phase Wrapping

Generally, in the real world, the experimental data is not as perfect as the simulated synthetic data. With the perfect DPC projections and phase images after 2D integration, we are trying to add phase wrapping to DPC images more close to real scenarios. Here we define a phase wrapping parameter "pixelSizeInMicrons" larger than π, like 10000. After 2D integration in Matlab, the phase image will look like,

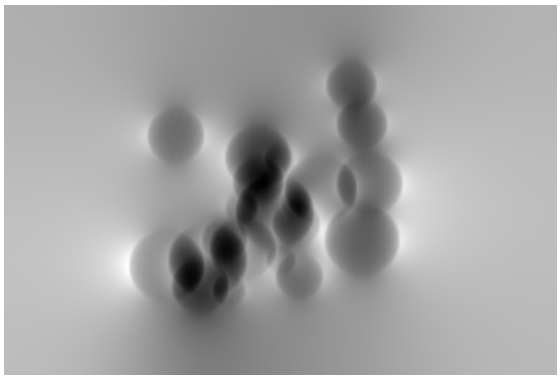


Figure 6.7: A single phase image at 0^o after adding phase wrapping

From the image above, obviously, the phase wrapping effect is strong. To explore the influence of phase wrapping to volume reconstruction, we pulled the phase projections into our TompPy/ASTRA jupyter notebook and obtained the phase volume rendering as

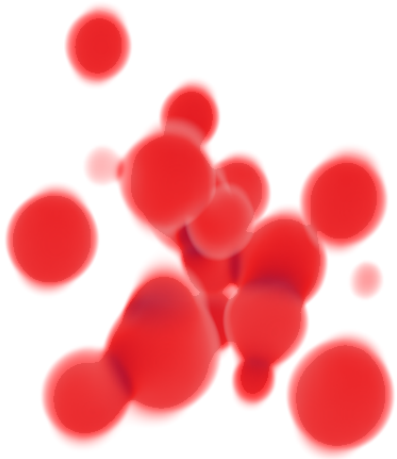


Figure 6.8: A 3D phase volume rendering with strong phase wrapping

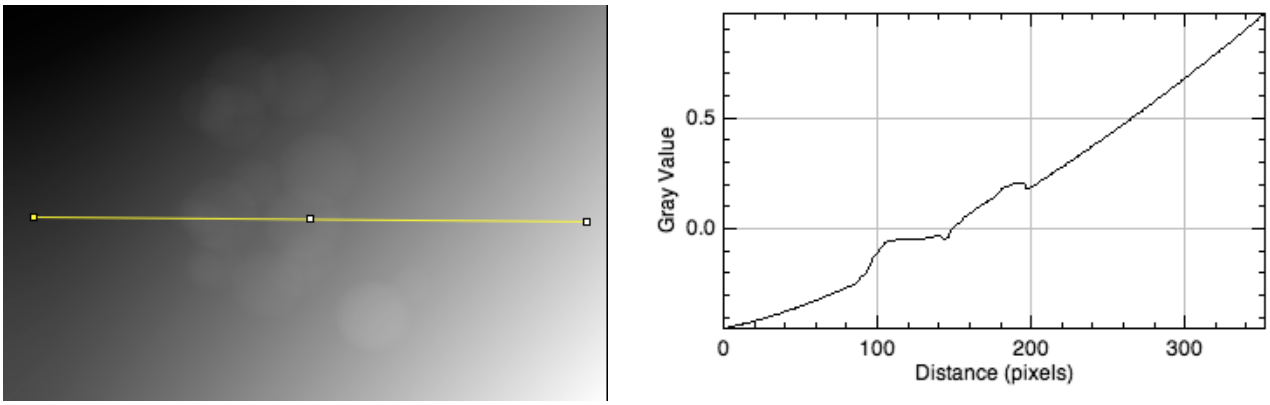
In the view of 3D volume rendering, phase wrapping has caused problems in the 2D integration process but we could still clearly see the sample structure.

6.4.2 Effect of Noisy Air

In most cases, the experimental data not only has phase wrapping problem but also has unbalanced or even tilt noisy air. Here we add a tilt noise to DPC projections in both vertical and horizontal directions. The tilt function is defined as,

$$slopePhase = 0.0003 \times \sqrt{range(columns)} \tag{6.13}$$

where $range(columns) = 1, 2, 3...384$.



(a) (b)

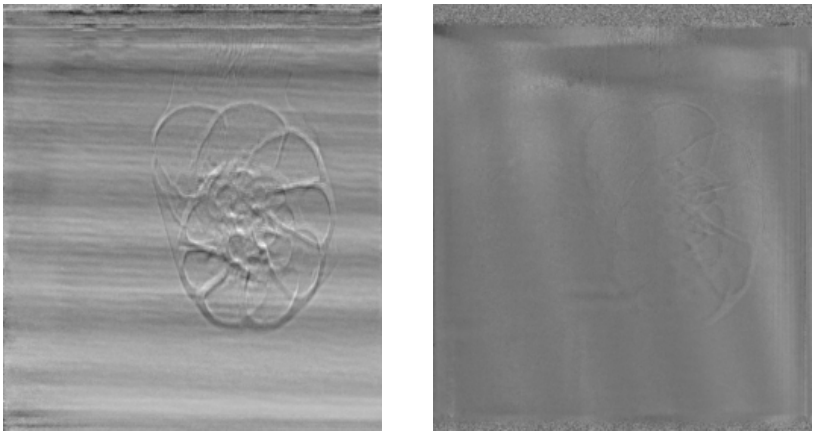
Figure 6.9: Synthetic phase projection image with noisy air at 0^o (left) and corresponded line probe (right)

From the lineprobe above, it clearly indicates the tilt noisy air does impact 2D integration in the experimental data. Thus, it is extremely necessary to clean noisy air before 2D integration and phase imaging reconstructions.

6.5 Workflow

Raw DPC projection images were obtained from X-ray interferometry experiment. In the experiment, both vertical grating and horizontal grating were conducted. To avoid a large storage, we downsized both vertical and horizontal projections. To smooth the downsized DPC images, Fast Fourier Transform (FFT) filter was applied to those images. With a smoother background, Harker O'Leary algorithm was used to integrate vertical and horizontal DPC projections, which would be used for volume reconstruction in TomoPy/ASTRA/Jupyter notebook with the reconstruction method of SIRT(Simultaneous Iterative Reconstruction Technique).

For the original DPC projections as shown in Fig. 6.10, the noisy air is a big affect to get insights into sample structures.



(a) (b)

Figure 6.10: Raw DPC projection images with (a) horizontal grating and (b) vertical grating

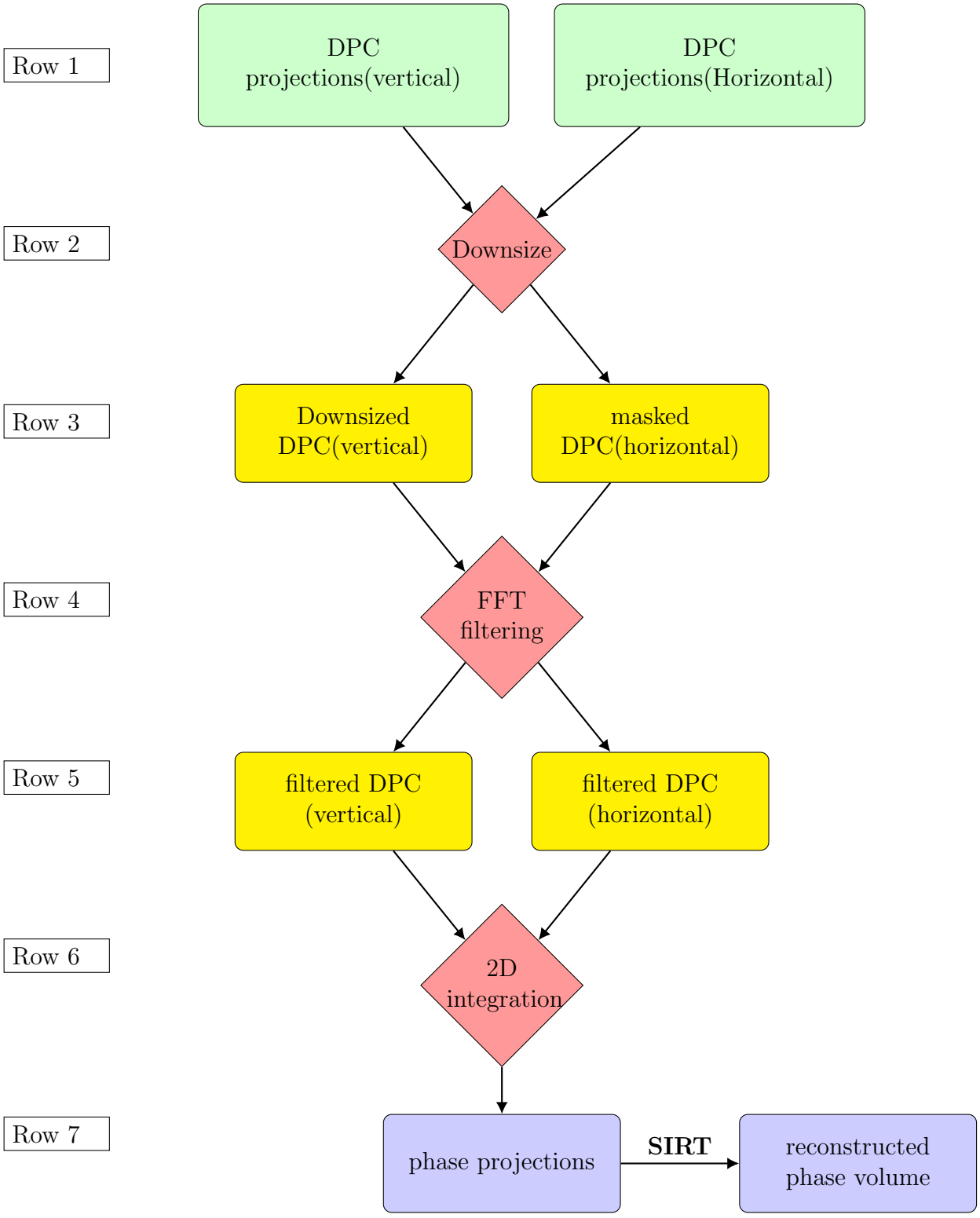
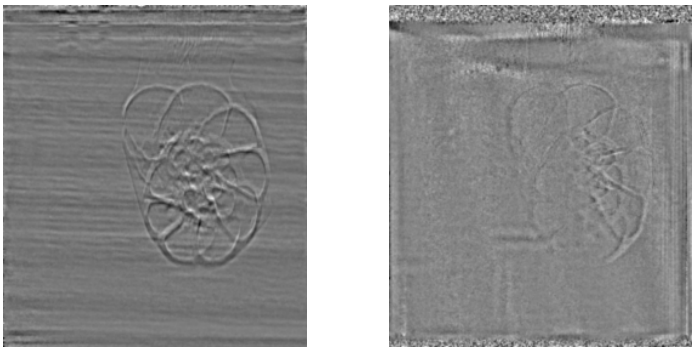


Figure 6.11: A scientific workflow for processing DPC projections with 2D integration Based on Harker-O'Leary algorithm

After downsize and FFT filtering, the fuzzy background with fringes were largely improved as show in Fig. **??**



(a) (b)

Figure 6.12: Integrated DPC projections with (a) horizontal grating and (b) vertical grating

Corresponding to the step in Row 6 6.11, 2D integration of DPC projections was completed with Matlab toolbox to produce a phase image 6.13. From the 3D view in MMATLAB, the air signals get weaker and fluctuate around zero with the exception of air values on the edge.

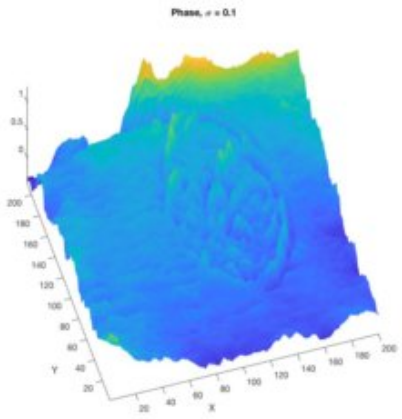


Figure 6.13: 3D view of Phase image in MATLAB

6.6 Results and Conclusion

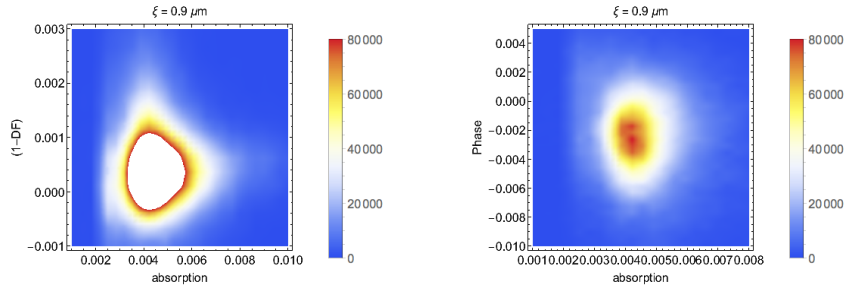
After reconstruction in TomPy/ASTRA/Jupyter notebook of the 2D integrated phase projections, we obtained a phase volume of sample foraminifera. In comparison to absorption volume and DPC volume in horizontal grating, the phase volume signals denote much

stronger 6.14c, which shows a great possibility to obtain more material information from phase volume. Herein, Harker O'Leary algorithm gets success in 2D integration of DPC images.

(a) (b) (c)

Figure 6.14: Screen shot of 3D masked volumes for (a) vertical grating DPC (b) horizontal grating DPC and (c) 2D phase integration in Dragonfly

As we know, there is a correlation between dark-field volume and absorption volume. The histogram 6.15a indicates the phase component in the sample. Similarly, we would like to explore the correlation between absorption volume and 2D integrated DPC volume in the same way. As shown in Fig. 6.15b, we could see in the middle part of sample foraminifera, both the absorption and phase density indicate strong signals; while, the shell of foraminifera shows weaker density. As a result, phase imaging could deliver more information of internal structures.



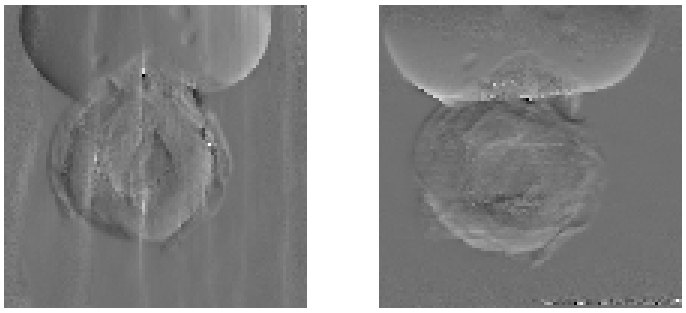
(a) (b)

Figure 6.15: Correlations of absorption volume versus (a)dark-field volume and (b) 2D integrated phase volume

6.7 Peppercorn Integration

Since there is a serious issue in grating motors at APS, the foraminifera DPC images has a bunch of fringes in the air which is hard to be eliminated. In this case, we conducted an interferometry experiment of object peppercorn in both horizontal and vertical directions with our own keck X-ray instrument.

With the same flowchart 6.11 but ignore the "downsize" step, we just pulled in the aligned peppercorn DPC projections.



(a) (b)

Figure 6.16: Aligned peppercorn DPC projection images in (a)horizontal and (b) vertical gratings

With Harker-O'Leary integration algorithm, we created phase images.

The noisy air of peppercorn data is a little bit tilt. Ideally, we expect the air is flat but in most real experiments, the air is noisy in DPC images.

As we know, the composition of peppercorn shell is almost uniform. Both of the absorption volume in Fig 6.18.a and the phase volume in Fig 6.18.c indicate a uniform structure, which means our 2D integration is successful to some extent. Meanwhile, the dark volume shows a strong scattering in the shell of peppercorn and air inside the sample.

If compared with foraminifera volumes in Fig. 6.15, peppercorn phase volume gives a much better performance in 2D phase integration. The reason is motor issues in instrument when collecting foraminifera data while in peppercorn experiment, motor issues have been largely improved to get a better quality of raw images without strong fringes. In conclusion, 2D integration of DPC projections in horizontal and vertical gratings generates a phase

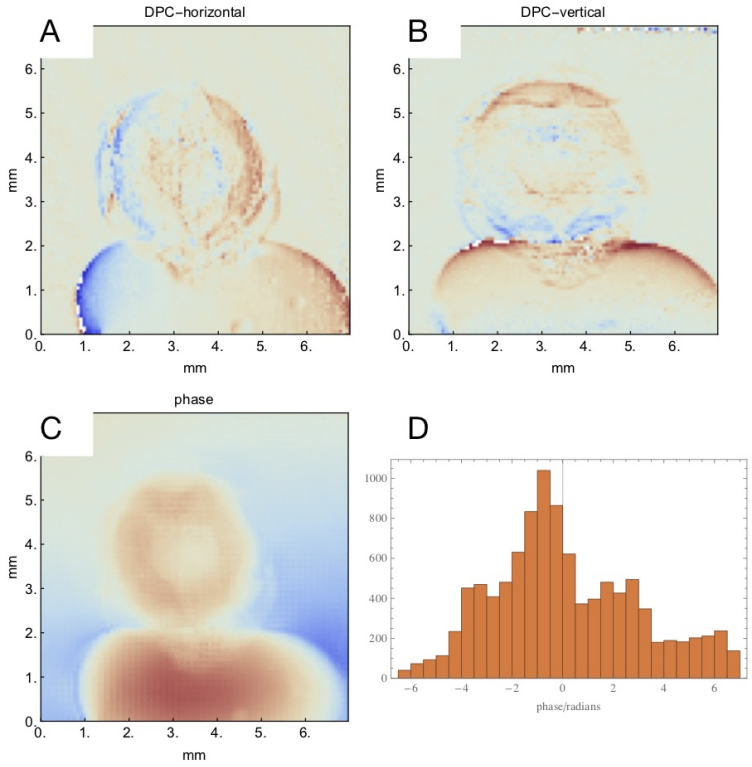


Figure 6.17: Peppercorn DPC projection images in horizontal and vertical directions as well as a phase projection image at 0^o degree and its corresponded histogram

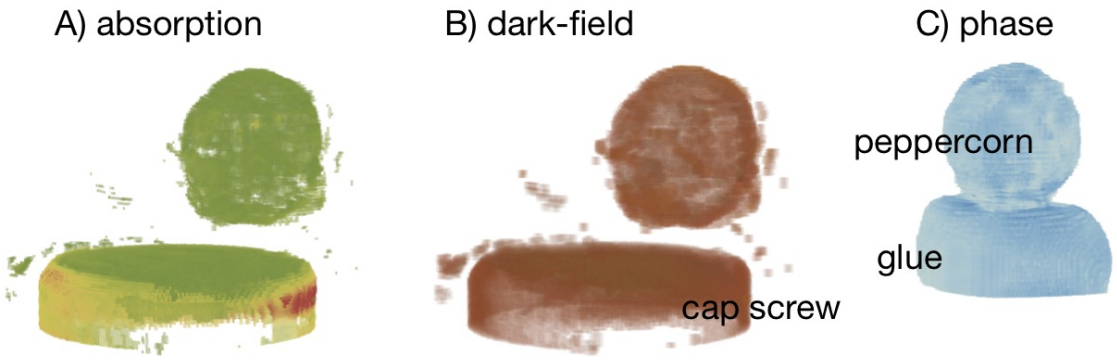


Figure 6.18: 3D volume renderings of (a) absorption, (b) dark-field and (c) phase images

volume which is quite helpful for internal structure exploration in samples.

6.7.1 Tunning Parameter Φ Analysis

The theoretical Φ in Momose's paper claims as [10],

$$\Phi(x,y) = \frac{2\pi}{\lambda} \sum \delta(x,y,z)dz \tag{6.14}$$

In our keck instrument, the experiment of peppercorn was conducted at the energy of $26kev$, which is equal to wavelength $\lambda = 4.77 \times 10^{-11}m$. Value δ is given in Momose's paper which is $\delta = 5.0 \times 10^{-7}$ for material polystyrene. The pixel size of instrument is around 69 microns and the thickness of sample is about 4.8 mm. With the equation (6.14), we calculated $\Phi = \frac{2\times 3.14}{4.77\times 10^{-11}} \times 5.0 \times 10^{-7} \times 69 \times 10^{-6} = 4.54$. However, our calculated $\Phi \approx 0.05 \sim 0.06$ which indicates a discrepancy between theoretical Φ and experimental Φ. Our future work will focus on the inconsistence of both two Φs.

6.8 References

[1] Atsushi Momose. Recent advances in x-ray phase imaging. *Japanese Journal of Applied Physics*, 44(9R):6355, 2005.

[2] P. O'Leary, M. Harker, and P. Zsombor-Murray. Direct and least square fitting of coupled geometric objects for metric vision. *IEE Proceedings - Vision, Image and Signal Processing*, 152(6):687–694, Dec 2005.

[3] Paul O'Leary and Matther Harker. An algebraic framework for discrete basis functions in computer vision. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, ICVGIP '08, pages 150–157, Washington, DC, USA, 2008. IEEE Computer Society.

[4] Matthew Harker and Paul O'leary. Regularized reconstruction of a surface from its measured gradient field. *J. Math. Imaging Vis.*, 51(1):46–70, January 2015.

[5] Matthew Harker and Paul OLeary. Direct regularized surface reconstruction from gradients for industrial photometric stereo. *Computers in Industry*, 64(9):1221 – 1228, 2013. Special Issue: 3D Imaging in Industry.

[6] R. T. Frankot and R. Chellappa. "a method for enforcing integrability in shape from shading algorithms". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988.

[7] R. T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, Jul 1988.

[8] M. Harker and P. O'Leary. Least squares surface reconstruction from gradients: Direct algebraic methods with spectral, tikhonov, and constrained regularization. In *CVPR 2011*, pages 2529–2536, June 2011.

[9] Matthew Harker, Paul OLeary, and Paul Zsombor-Murray. Direct type-specific conic fitting and eigenvalue bias correction. *Image and Vision Computing*, 26(3):372 – 381, 2008. 15th Annual British Machine Vision Conference.

[10] A. Momose. "recent advances in x-ray phase imaging". *Japanese Journal of Applied Physics Part 1-Regular Papers Brief Communications & Review Papers*, 44(9A):6355–6367, 2005.

Appendix A
VisTrails Tutorial

- 1. Prepare the Mathematica script to accept arguments from command line and export figures to PNG files. It should run like this:

  ```
  /path/to/MathKernel -script scriptname.m <arguments>
  ```

- 2. Construct a simple VisTrails workflow to run this Mathematica script. The modules include:

  - 2.1 Multiple String modules: as user-input parameters
  - 2.2 A PythonSource module: connect to parameters and assemble a bash script (e.g. a "run-script" file) to run the Mathematica script, make sure all path set correctly, so you can just run this command on remote host:

    ```
    $ ./run-script
    ```

  - 2.3 A SubmitJob module: connect to PythonSource module, get the runscript and copy to the server
  - 2.4 A Queue module: specify your remote host
  - 2.5 Optional DownloadFile, DownloadDirectory modules: to download data from server to local
  - 2.6 Optional ImageViewerCell modules: to show downloaded images in spreadsheet

- 3. The Bullet VisTrail Workflows Overview This workflow serves as a simple example of using VisTrails software to manage Tomography data processing workflow.

  A complete tomography experiment usually produce many datasets (collected using different settings, or from different instrument) with large size (several GitBytes). The data processing workflow includes raw data checking, data reconstruction, and a wide range of data analysis tasks(visualization, segmentation, skeletonization, surface generation, measurement, histogram, etc.). The data processing code are developed by community using Mathematica, Matlab, Fortran, C/C++, Python, etc. We also want to run data processing jobs in parallel submitted to HPC resource.

  We choose to use VisTrails workflow to create a unified developer and user interface for Tomography data processing. VisTrails provides functions to make this happen:

  - 1. **Provenance**: development process are fully tracked, history can be tagged and versioned and be revisited again with preserved settings.
  - 2. **Parameter exploration**: explore a parameter with a range of values, with results presented side-by-side in spreadsheet for easy comparison.

  - 3. **Wrap commands into modules** : write a python source, or use CLT command line tool to wrap executables and run scripts as a module. Once modules are constructed, users can switch different toolsets by drag and connect different modules to the current pipeline, the pipeline runs the same way.
  - 4. **Submit remote jobs**: The tej tool provides a way to start job on any remote server through SSH, associate it with an identifier, and monitor its status. When the job is complete, it can download the resulting files through SCP.
  - 5. **Generate simplified user interface**: mashup creates a user interface composed of chosen parameters and a result view.

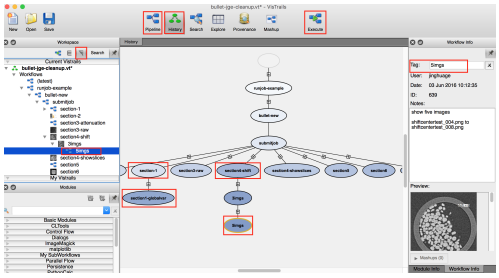
- 4. Workflow structure The original Mathematica notebook include six sections:
  - 1. **Initialization**: convert to script section-1.m
  - 2. **Attenuation at the first rotation angle**: convert to script section-2.m
  - 3. **Attenuation**: convert to script section-3-raw.m and section-3-attenuation.m
  - 4. **Make Sinograms**: convert to script section-4-shiftcentertest.m and section-4-showslices.m
  - 5. **Reconstruct the sinograms into slices**: convert to script section-5.m
  - 6. **3D Volume Plot**: convert to script section-6.m

  Following the instruction in Review, we have completed an example vistrail to run a bullet data reconstruction project. The complete vistrail include eight workflows, each run one .m script. Users should run these workflows in consecutive order to complete the whole process. This is because the later workflow depends on the output of previous workflow.

- 5. Navigate the versions If you cloned our repo, you can find the vistrail file at

  `/path/to/LSU-VisTrails/vistrail/bullet-cleanup.vt`

  Open this file in VisTrails. Click the treeview in workspace panel, and click the history tool to show history view in the main panel. You can see all of the workflow versions clearly in this setup. The screenshot below shows the version tagged 5img is chosen.

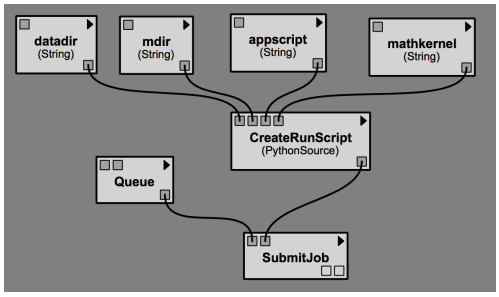


A screenshot of VisTrails GUI

After a version is chosen, click the pipeline tool will show the workflow pipeline in the main panel.

- 6. Global Variables Before we run a workflow, lets introduce the VisTrails global variable. Look back at the prevoius screenshot, notice in the version tree there is a section1-globalvar version underneath section-1 version.

  The section-1 workflow is very simple, in its pipeline there are three user-input parameters (String modules datadir, mdir, mathkernel) that are used in every workflow.

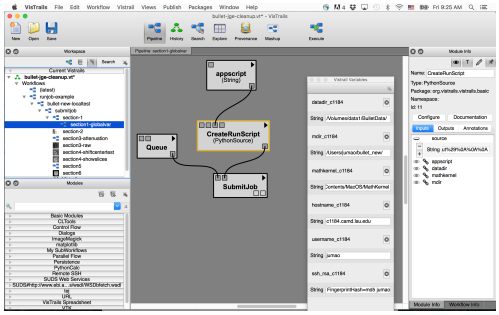


Pipeline in VisTrails workflow

  There is option to create global varible for them. (View->VisTrails Variables). Then you can delete the Sting modules and drag the global var to the port.

  Similarly, you can also create global vars for module Queue port hostname and username

  When use global vars, the pipeline look much cleaner. Below is screenshot:



A screenshot of pipeline with global variables

- 7. Run a workflow Now we show how to run a workflow and get its results. Either double click a version in workspace tree view, or go to history panel and select a version, then switch back to pipeline view. You should see this workflows pipeline modules.

  Here well use the 5img version under the section4-shift version as an example.

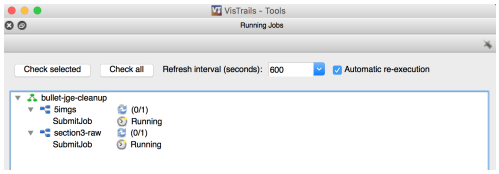
- 8. Change user-input modules Before you run this workflow, change some parameters to accommodate the data and script path, and your running computer info.

  - datadir(String): change to where the dataset structure is saved.
  - mdir(String): change to where the .m scripts are saved
  - mathkernel(String): change to Mathmatica kernel in remote computer. In Mac OS X, most probably its at : /Applications/Mathematica.app/Contents/MacOS/MathKernel

- Queue : set hostname and username values in input ports
- DirectorySink: set local directory for the downloaded directory

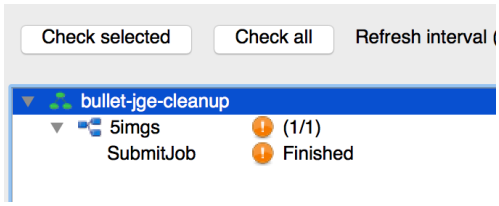
  After the modules are setup for your run, click Execute in toolbar. The runningjobs panel will showup, telling you that your job is running. Suppose your job will run for a while, you can go do something else for now.

The runningjobs panel show the job status (View->Running Jobs)



A screenshot of job status in VisTrails

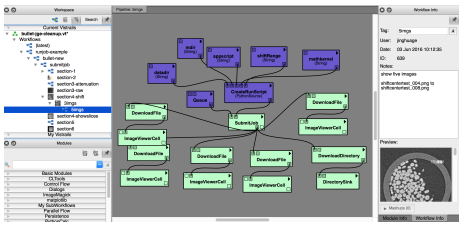
- 9. Come back and check results
  When you come back you should see that your job is finished.



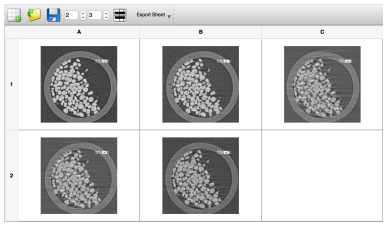
A screenshot of job progress in VisTrails

If you click Execute again and nothing has changed, VisTrails will go fetch the cached results and show them in spreadsheet. See the below screenshots for executed pipeline and spreadsheet.

The purple color coded modules means nothing changed and use cached result.



(a)



(b)

(a) A screenshot of VisTrails modules with different colors (b) Reconstructed bullet slices in spreadsheet

- 10. Troubleshoot
  If it happens that you want your jobs to run again instead of using the cached results. you can go ahead delete all job status in runningjob panel, and delete the jobs Stage folder as well.

Appendix B
TomoPy Tutorial

This appendix will show users how to use our Jupyter notebook step by step in specifics. Meanwhile, videos are available in our YouTube channel .

Jupyter Notebook directory:

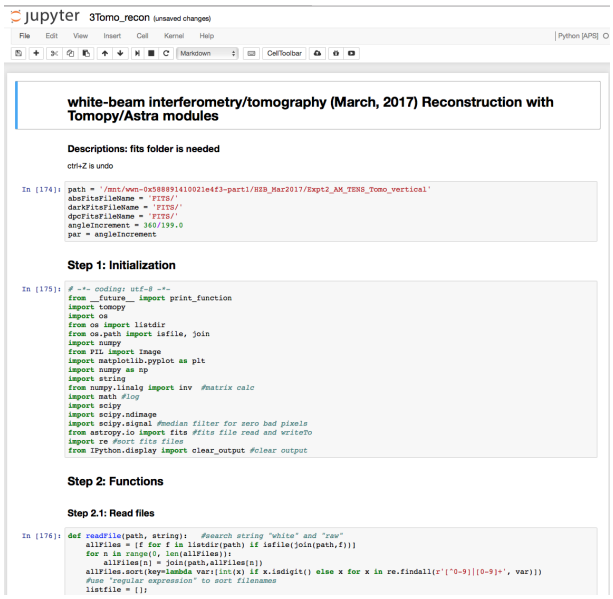
$ ssh 130.39.161.20 contact [1] for username and password

Jupyter is a user-friendly editor for Python programmers. Our Jupyter notebook is based on Python platform Anaconda2. In this way, multiple kernels could be working at the same time.

Open a terminal and type:

$ jupyter notebook

then you can start on Python coding.



Jupyter notebook user interface

The routine for Dogbone reconstruction is thought raw data → projection → reconstructed 3D volume with TomoPyASTRA toolbox

If you need more details, our scripts is open-source in Github [2].

Here, we use tomopy.gridrec for absorption reconstruction and ASTRA-SIRT for dark-field image reconstruction.

For tomopy.gridrec,

```
recon = tomopy.recon(dpcProj, theta, center=rot_center, algorithm='gridrec')
```

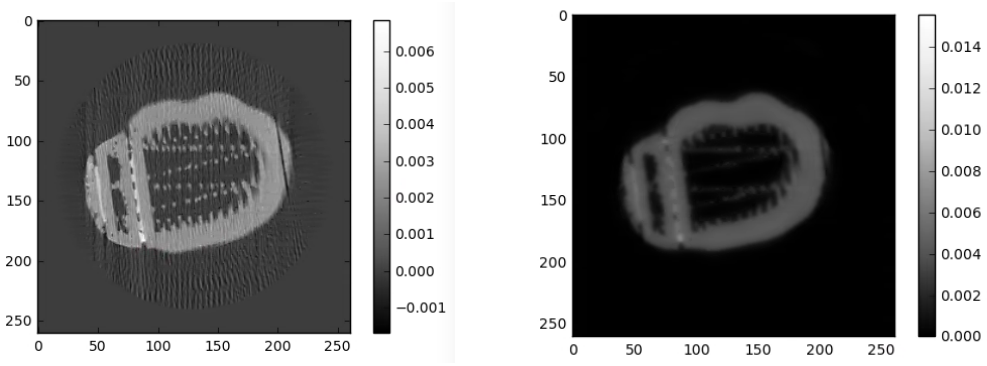
[1] jyuan4@lsu.edu

[2] https://github.com/jyuan4/Jupyter-TomoPy-ASTRA

```
recon = tomopy.circ_mask(recon, axis=0, ratio=0.85)
```

For ASTRA-SIRT, NVIDIA GPU based core need be installed before running ASTRA SIRT command lines,

```
#tomopy.astra?
extra_options ={'MinConstraint':-0.5}
options = {'proj_type':'cuda', 'method':'SIRT_CUDA',
'num_iter':40,'extra_options':extra_options}
darkRecon = tomopy.recon(darkProj, theta, center=rot_center,
algorithm=tomopy.astra, options=options)
darkRecon = tomopy.circ_mask(darkRecon, axis=0, ratio=0.85)
```



(a) (b)

Reconstructed slices of 'nose down' bunny with (a) 'Gridrec' in TomoPy package and (b) 'SIRT' in ASTRA toolbox

Appendix C
Dragonfly Tutorial

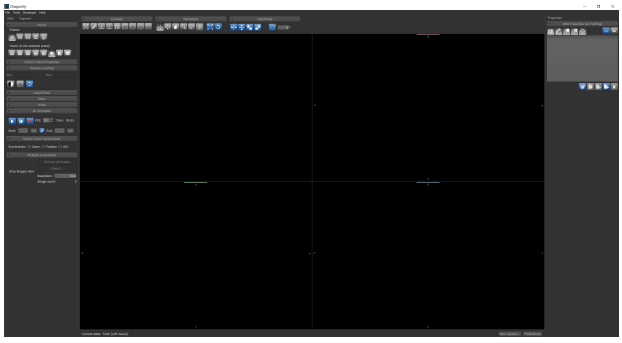
This appendix contains detailed notes for using Dragonfly to perform the analysis shown in the flowchart, Fig.6.11. A similar workflow uses Mathematica and is described in Appendix D This Dragonfly workflow can be applied to any two-volume data set with the following attributes:

- The two volumes share a coordinate system. This is naturally the case for Talbot-Lau stepped grating interferometry.
- The air values for attenuation and (1–dark-field) are near zero. Samples values are greater than zero.
- The attenuation volume has good image contrast and is chosen for mask creation.
- The structures to be analyzed in the workflow are not required to be aligned with a Cartesian axis, although nearly linear structures are desired for both line probes and mask-based analysis such as the Mathematica workflow and the bone analysis macro.

The row numbers below are defined in the flowchart in Fig. 6.11. Import – Row 1
The attenuation and dark-field volumes from a neutron interferometry/tomography experiment are normally created in HDF5. In this work, reconstruction was done in a Jupyter/-TomoPy/ASTRA workflow.

Cropping – Row 2
In ImageJ, two HDF5 volumes were cropped identically into volumes of __×__×__ and saved as real-32 TIFF volumes.

Read in .tif files - Row 3
Here, the Dragonfly workflow begins.

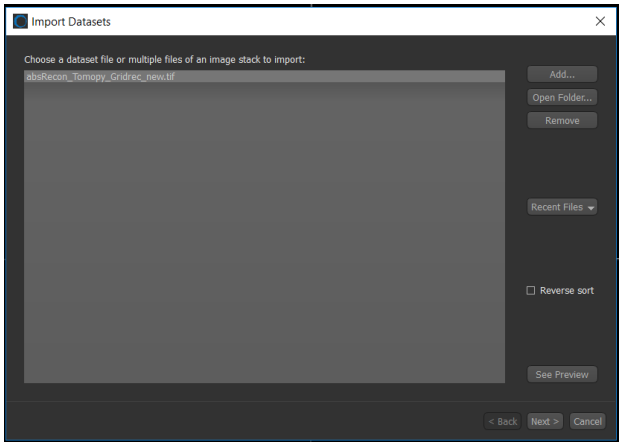


Dragonfly default display. Note the as yet empty workspace in the upper right.

- Files → import image files → Add

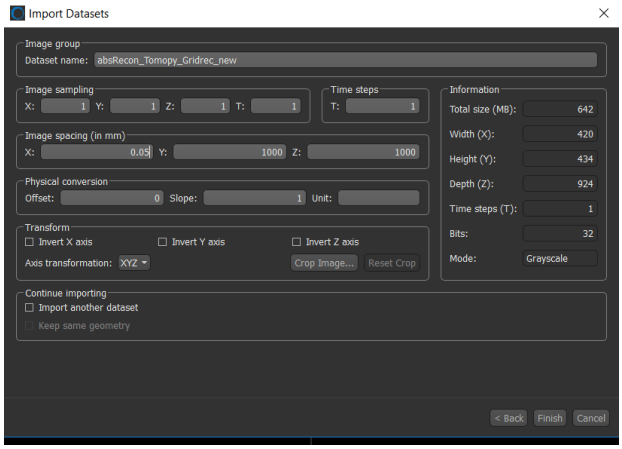
- Note the population of the workspace panel with the attenuation and dark-field volumes.

Read in .tif files - Row 1
Get stared with Dragonfly.



Dragonfly import files

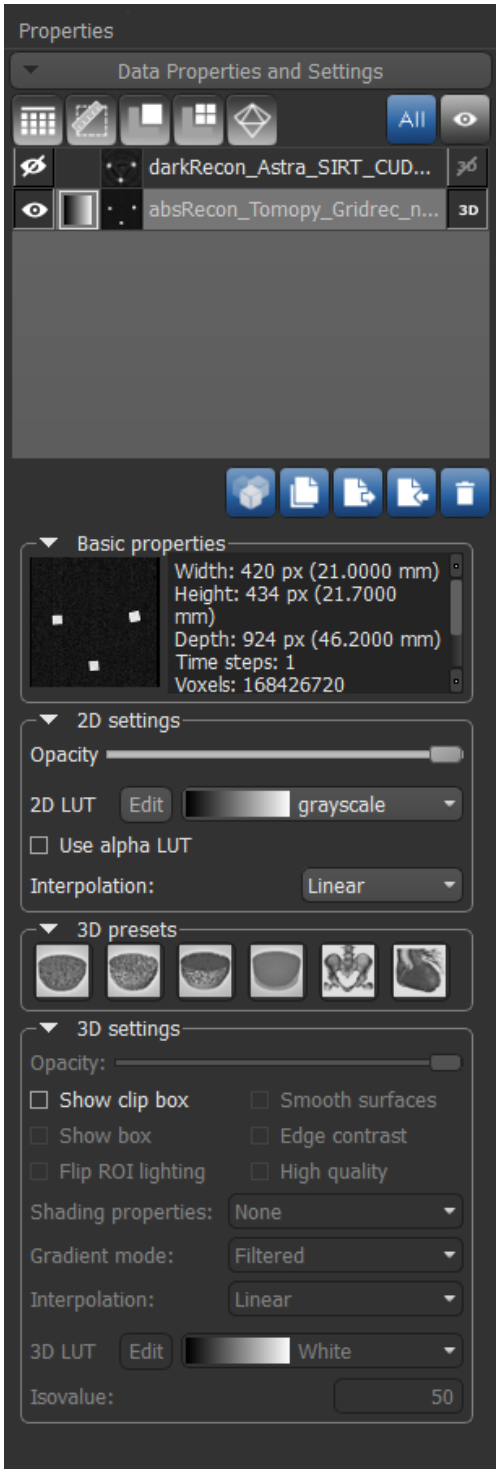
- Next → Image spacing (in mm) → X, Y, Z: 0.05 → Finish



Dragonfly change parameters for input image files

If you read in absorption image file, then do the same thing with dark-field image. Now, these two datasets are read in Dragonfly session and save.

Next, control 2D/3D setting in the right side panel for each dataset.

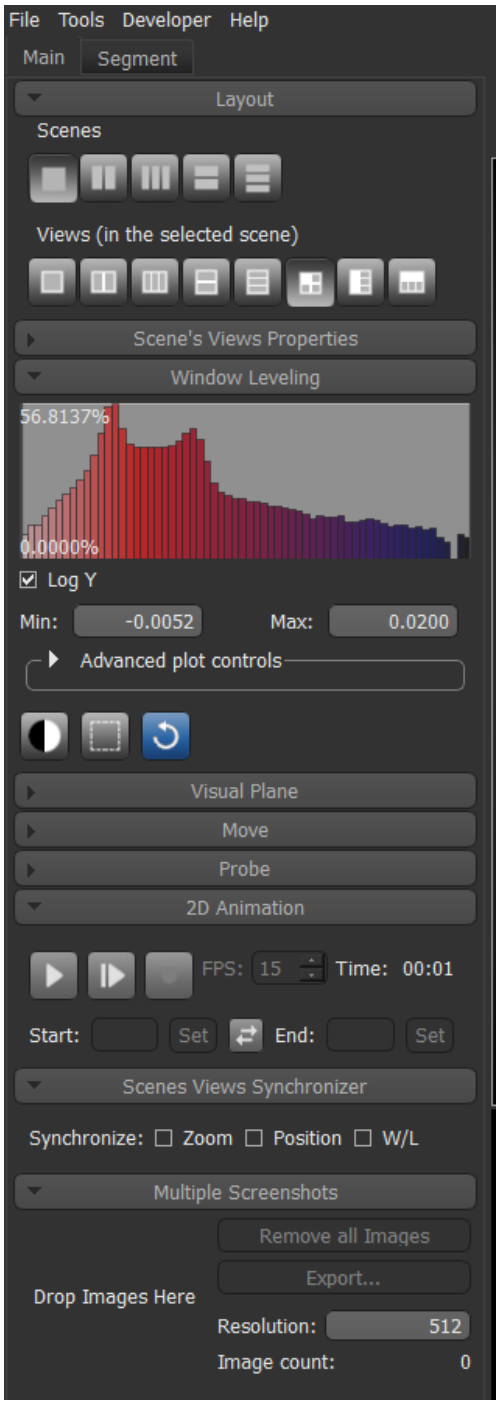


Dragonfly properties panel

- click absRecon file and make absorption image viewable → choose 2D image → edit

2D setting: Cyan → choose 3D image → edit 3D setting: blue red

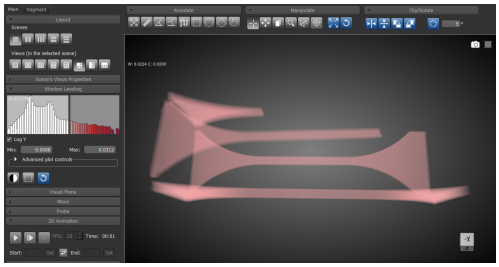
Then, go to the left panel: Main and Segment



Dragonfly main panel

- Window Leveling → select 3D image → Drag histogram → 3D volume is shown

In the 3D volume above, "Annotate", "Manipulate", and "Flip/Rotate" are used to control the position, shape, size, zoom in/out etc. of 3D volume.



Dragonfly 3D volume

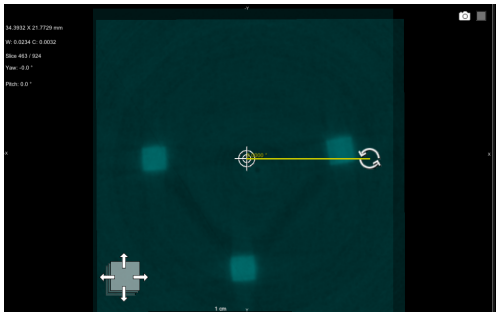
And do the same thing with dark-field image, save.

Alignment - Row 3

Register and Explanation in Dragonfly [1]

If you look at the properties panel: basic properties. We can find the size of abs and dark are different, so alignment is needed in this step.

- right click darkRecon file → Data Registration → select Translation → Apply
- left panel: Main → Move → select Displace (most left icon) and "Dynamic refresh" → right panel: 2D settings, Opacity → adjust Opacity and begin alignment manually



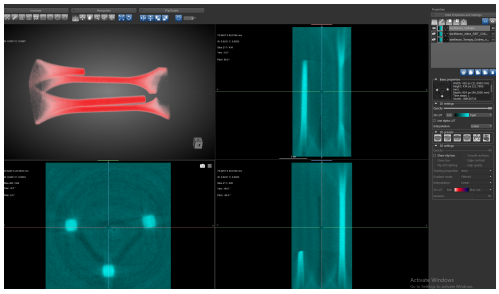
Dragonfly alignment

- right panel: Properties → right click darkRecon → Resample shape → Rename and click OK

Now, "darRecon_shape and "absRecon" images are the same size.

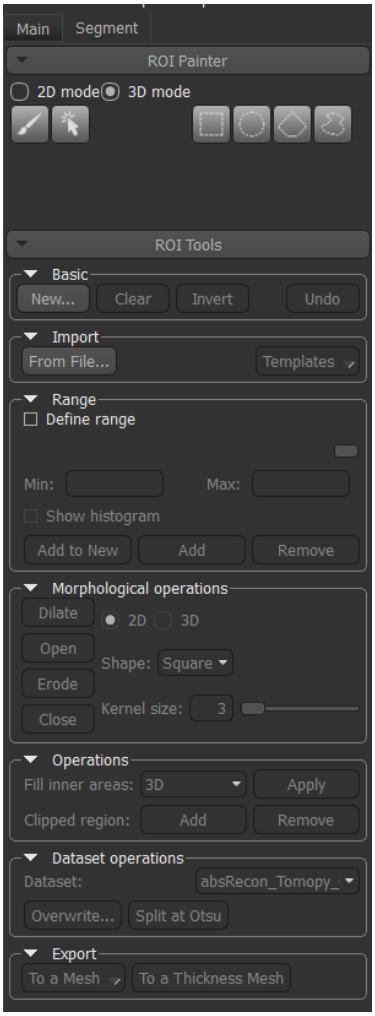
For the darkRecon_shape image, adjust 2D/3D settings first, save session.

[1] `https://youtu.be/r0TRTLbav28`



Dragonfly image reshape

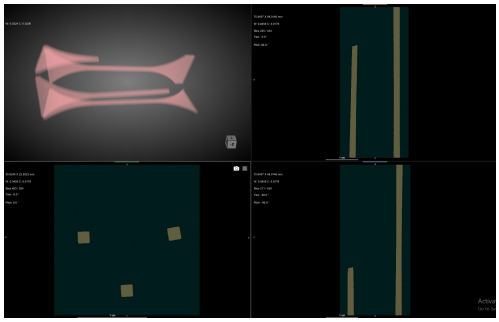
Binarization - Row 4
Segmentation in Dragonfly [2]



Dragonfly segment control panel

[2] https://youtu.be/Wh0xa7Mym_4

- left panel: Segment → choose absRecon in the right panel: Properties → left panel: segment, New → Name: foreground → select Define range → adjust two range bars manually → Add to New → Dilate → Apply → Add → Invert → Data properties: Overwrite both absRecon and darkRecon_shape as value 0



Dragonfly binarization(segmentation)

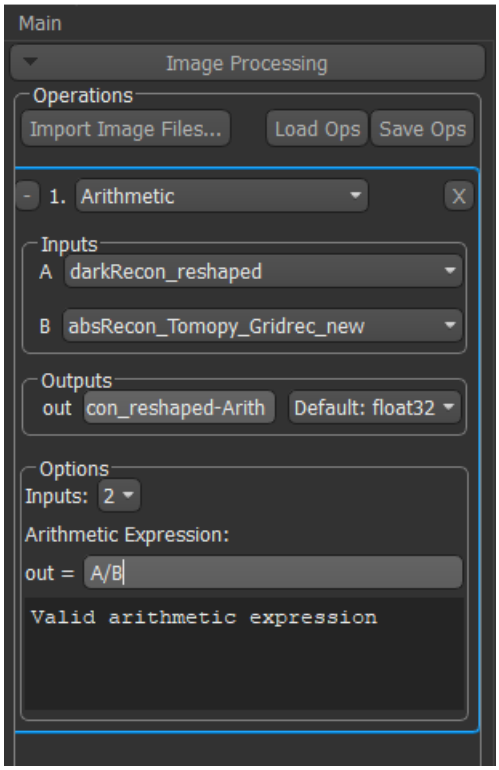
Make mask for A and DF (filter)- Row 5
After binarization, export image files and open imaging toolbox (or Macro written in Python) [3]

- right panel: Properties → right click absRecon image → Export Images... → check with Image (32-bit real)

Arithmetic (DF/A) - Row 6
image toolbox in Dragonfly

- Tools → Image Processing Toolbox → 1.Operation: Arithmetic → Inputs: 2 → A: darkRecon_reshaped and B: absRecon → calculate A
  B → compute All Previews and export masked dark-field volume

[3] `https://youtu.be/E1gx9dLVphA`



Dragonfly arithmetic: DFA

Line Probe and Bone Analysis - Row 7
Main panel in Dragonfly and Python scripts in console [4]
Bone analysis (Macro)

[4] https://youtu.be/K6EmUbSynok

Appendix D
Mathematica Tutorial

Step by step instructions on bullet processing with the Mathematica program, and then the VisTrails version. This may be so long and detailed section. It won't be published in manuscript, but will be in appendix or our website.

- Step 1. Initialize.
  The path to the raw image files (32 MB?) is defined relative to the notebook. The paths to intermediate folders for figures, attenuation data files, sinograms, slices, and volumes. Image processing functions are defined for the following steps.
- Step 2. Attenuation at the first rotation angle
  - 2a. Open Beam. Image taken with neutrons on and sample removed. This measures I_o.
  - 2b. Dark. Image taken with neutrons off. This measures detector bias.
  - 2c. Read 0 degree raw and make attenuation image. Import 0.0 degree bullet raw data image and calculate attenuation by using formula $log\frac{openbeam/600-dark/100}{raw/140-dark/100}$.
  - 2d. Normalize the intensity values based on air region. After normalizaiton, the air region should have an average intensity value of 0. The object region is forced to have an constant average intensity. We use a linear equation for the normalization: corrected intensity = uncorrected intensity - offset.
  - 2e. Rotate the attenuation image by 90 degree and save as *.fits. FITS file format reads nicely into ImageJ either as single image or a sequence of images.
  - 2f. Set grayscale plot limits (hardwired now, in the future should have VisTrails controls), replot and save as *.png
- Step 3. Attenuation
  - 3a. Get filenames of all raw images. Read bullet raw images in the order of angle list.
  - 3b. Make a short table of rotation angles and corresponding filenames. Concatenate raw images file name and corresponding rotation angle as a new file name.
  - 3c. Read all *text files and make a movie. We read all "Bit 16" raw images and make a movie for any OS system (Mac OSX/Windows/Unix). From the movie, we learn that there are two bad frames.

  - 3d. Red all raw images and make all attenuation files. Save as FITS (about 4 min on laptop). Here, we use parallel computing to run our loops. Instead of making one attenuation file, we make all attenuation files for all raw images as above steps and save them as FITS files finally.
  - 3e. Replace bad FITS files at #22 and #46 with the average of the neighboring data files. From the movie in step 3c, we learn there are two bad frames: #22 and #46. In the current step, we try to replace these two bad frames by averaging the neighboring (the previous and the next) data files.
  - 3f. Read all absorption FITS and Create *.png.
  - 3g. Read all *.png and make a movie (about 10s)

- Step 4. Make sinograms (2 min)
  - 4a. Read all attenuation FITS and make allSinograms. From all attenuation FITS images from above, we make 3D allSinograms with [NY, NX, NZ], where each image size is NY×NX and the angle is NZ.
  - 4b. Find best center of rotation. This step is best shiftvalue exploration: We estimated the centering shift value range is [-1,1] with stepping = 1 based on 0 degree and 180 degree sinograms.
  - 4c. Shift all sinograms and plot a few reconstructions. Now, we use the determined best centering shift value from above step and then shift all sinograms with that best centering shift value.
  - 4d. Save sinograms as FITS files. based on step 4d, we saved all shifted sinograms as new FITS files.

- Step 5. Reconstruct the sinogram into slices (1 min) Save slices as FITS and save volume as HDF5.
  - 5a. Read all sinograms and reconstruct into slices. With *InverseRadon* (it gives the inverse discrete Radon transform of image.) in Mathematica, we reconstruct all sinograms into slices.
  - 5b. Read all slices and save as one HDF5 volume. with reconstructed slices from above step, we save all slices as a single HDF5 volume, which is better for view.

- Step 6. Store all slices into one HDF5 file
  - 6a.Make a 3D and display. Read HDF5 file and display 3D volume bullet object.

Functions for local pixel error correction Raw, open, and dark images. Find, order, and import

There are 8 bunny date sets. Here, we take W2bunny, Horizontal grating bunny as an example.

1st run: tiff, FITS and 2nd run: w2_tiff. 2 degree increment, horizontal gratings with 0.5 micron grating step. 3rd run: w3_tiff, 23_FITS, 1 degree increment

- Step 1. Define functions
  - 1a. functions for reading file, finding number of groups of reference files and sample files. For reference file? sample file?

    funcReadPilatusFile[filename_] This will read data from a Pilatus integer-32 TIFF file. The standard Mathematica Import[filename] cannot read integer-32 TIFF.
    funcFindReferenceTIFFfiles[pathTIFF_] Finds files based on ".white." and orders the list based on the image sequence number.
    funcFindSampleTIFFfiles[pathTIFF_] Finds files based on ".raw." and orders the list based on the image sequence number.
    funcFindReferenceGroups[] Based on sequence number, finds the grouping of the reference files.
    funcFindSampleGroups[] Based on sequence number, finds the grouping of the sample files.
  - 1b. functions for interferometry
  - 1c. functions for finding correct files for a given rotation angle
  - 1d. plot functions for interferometry results. For convenient, we defined a few graphical functions in the next steps.
  - 1e. functions for tomography.
- Step 2: Paths, filenames and grouping: Set interferometer steps and period
  we created paths for TIFF, HDF5, FITS, Slices, Figures, Volumes and Sinograms files.
- Step 3: For any angle, process reference and sample interferograms.
  - 3a. initialize vectors used for the calculation.
  - 3b. set angle for calculation, find the correct filenames. Here, we randomly picked the first sample angle for test.
  - 3c. calculate transmission, visibility, and phi for reference and sample. Just call functions from Step 1.
  - 3d. correct for bad pixels: reference transmission, sample transmission and reference visibility. We find bad pixels and correct them.
  - 3e. plot transmission, visibility, and phi for reference and sample.
  - 3f. calculate absorption, differential phase contrast, and dark-field. Where absorption is based on Beer's law: absorption $= -log\frac{sampletransmission}{referencetransmission}$.
    differential phase contrast = sample phi - reference phi.
    dark field $= \frac{samplevisibility/referencevisibility}{sampletransmission/referencetransmission}$
  - 3g. plot absorption, differential phase contrast and dark-field.
  - 3h. plot doseROI. what is doseROI?

- Step 4. for all angles, calculate absorption, DPC, and dark-field projections from the interferograms.
  - 4a. initialize vectors used for the calculation
  - 4b. for all angles, calculate absorption, DPC, darkfield and percent visibility (sample visibility percent = $100 \times \frac{samplevisibility}{sampletransmission}$) and store HDF5, FITS. Here, we use parallel computing to print rotation angle, date, and file name sequentially.
- Step 5.Centering
  - 5a. Read 0 and 180 degree absorption FITS and make allSinograms.
  - 5b. find best center of rotation. Based on 0 degree and 180 degree absorption images, we estimated centering shift range is [-1, 1] with stepping = 1.
  - 5c. shift allsinograms and plot a few reconstructions. based on above best centering shift value, we shift all sinograms and test initial reconstructions.
- Step 6. Make sinograms for ASTRA (FBP or SIRT with parallel beam geometry)
  - 6a. read 0 to 180 degree absorption HDF5 and make allSinograms; save *.bin
  - 6b. absorption: shift the sinograms, and save as bin and HDF5. With larger field of view, remove streaks, apply air offset
  - 6c. dark-field
  - 6d. DPC
  - 6e. DPC_corr
- Step 7. Volume: customize for the sample - slow and big 3D volume (ingored in VisTrails)

The Mathematica notebook is uploaded into Overleaf in `Appendix\HZB_lineprobe_Expts2-6-8.nb`.

This is a sketch of the Mathematica notebook used for the line probe through a dogbone. This notebook serves as a guide for the DragonFly macro. The macro does not need to follow exactly the same calculation. For example, the line probe values in Mathematica and DragonFly will probably be calculated by different routes. Also, the Mathematica line probe plot superimposes the results for three different tomography runs; only one tomography run is processed with DragonFly.

Here are some observations about the Mathematica notebook that may be relevant to the DragonFly macro. (1) Multiple samples: Neutron tomography time is expensive, so whenever possible, multiple samples are imaged at once. Then, a "good" sample appears. (2) DF and (1-DF): The dark-field image in projection view has air=1 and sample $\in [0, 1]$ (ignoring noise). For dark-field in tomography view, we work with (1-DF) so air=0 and sample $\in [0, 1]$. (3) Alignment to cartesian axis system: The Mathematica notebook does not go this route, but it is probably the best option for DragonFly. (4) Plotting (1-DF)/A:

The dark-field as (1-DF) shows scattering from the sample, both internal cracks and surface roughness. The attenuation, A, shows where the sample is. The normalization with (1-DF)/A, and the masking, is a step towards ignoring surface roughness. (5) Masks: The Mathematica notebook has three masks. There are elliptical masks to separate the four different samples. There is an eroded by 5 times mask from the attenuation volume; this selects the internal structure. There is a shell mask—difference between a dilation and the erosion-5—to select for surface cracks. (6) Divide by zero: Mathematica notebook uses a Matlab "find" operation (called "Position" in Mathematica) to select only those list elements with non-zero values for attenuation. For AM work, line probes extending past fracture points and into air will be common, so please start avoiding divide by zero errors.

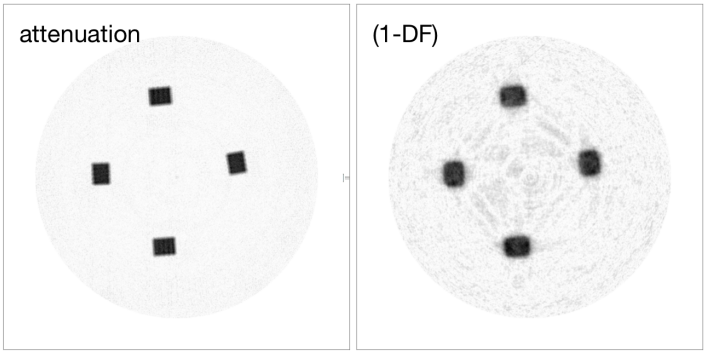
The Mathematica notebook is uploaded into Overleaf in `Appendix\HZB_lineprobe_Expts2-6-8.nb`. Step 5 is slow (several minutes on a laptop) and some code from Step 5 is shown in Fig. **??**. To select different options in the program such as shell or erosion-5 mask, lines are un/commented. An overview of all steps is shown in Fig. **??**.

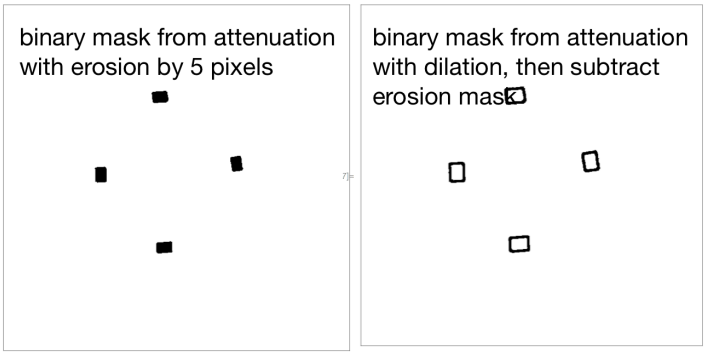
An overview of the HZB_lineprobe_Expts2-6-8.nb Mathematica notebook.

Fig. **??** shows slices of the attenuation, A, the (1-DF), erosion-5 mask, and shell mask.

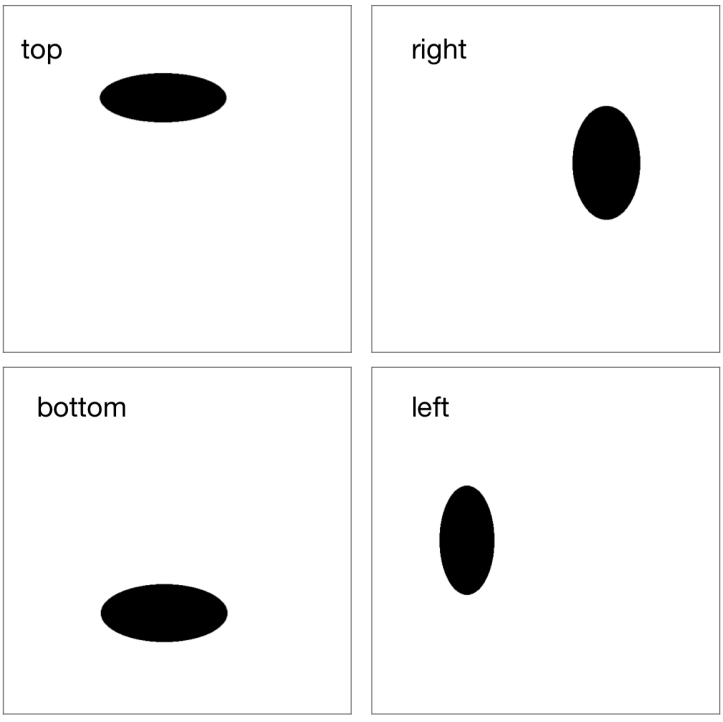
Fig. **??** displays the masks used to select the four samples. The top and left are from the fractured dogbone; the top ellipse is for the short fractured sample. The 75% stressed sample is opposite the short fractured sample; the stressed sample is a few mm longer than the pristine sample which is opposite the long fractured sample. **Fiducial marker:** The end of the short fractured sample is the 0 mm marker in the line probe plots, Fig. **??**, **??**, and **??**.

Fig. **??** shows some code used to process the attenuation and (1-DF) values of the stressed dogbone. This code processes one slice; the code is embedded in a loop over all slices of interest in Step 5. The first line uses a mask to select for the dogbone on the right side of Fig. **??**. The DragonFly line probe macro probably doesn't need this step as DragonFly can define a line probe with start-stop coordinates. **Statistics:** The





Slices from four HDF5 volumes for attenuation, (1-DF), erosion-5 mask, and shell mask.



mask_dogbones

Mathematica code returns both the mean and the standard deviation, provided at least two non-zero voxels are found in the slice.

```
index = indexRight = Position[sliceMask * maskRight, p_?(# > 0 &)];
Dimensions[indexRight]
dataAttenuation = If[Length[index] > 0,
    Table[Module[{r, c},
      {r, c} = index[[i, All]];
      sliceAttenuation[[r, c]]], {i, Length[index]}], {}];
dataDarkField = If[Length[index] > 0,
    Table[Module[{r, c},
      {r, c} = index[[i, All]];
      sliceDarkField[[r, c]]], {i, Length[index]}], {}];
indexNonZero = Position[dataAttenuation, p_?(# > 0 &)] // Flatten;
dataAttenuation = dataAttenuation[[indexNonZero]];
dataDarkField = dataDarkField[[indexNonZero]];
ratioDFvsAttenuation = If[Length[dataAttenuation] > 2, dataDarkField / dataAttenuation, {0, 0}];
dataAttenuation = If[Length[dataAttenuation] > 2, dataAttenuation, {0, 0}];
dataDarkField = If[Length[dataDarkField] > 2, dataDarkField, {0, 0}];
{Dimensions[ratioDFvsAttenuation], Dimensions[dataAttenuation], Dimensions[dataDarkField]}
dataRight = Append[dataRight,
  {indexSlice, Mean[ratioDFvsAttenuation], StandardDeviation[ratioDFvsAttenuation],
   Mean[dataDarkField], StandardDeviation[dataDarkField],
   Mean[dataAttenuation], StandardDeviation[dataAttenuation]}]
```

```
{571, 2}
```

```
{{571}, {571}, {571}}
```

```
{{650, 2.3649, 0.157288, 0.0088877, 0.00047255, 0.0037646, 0.000160755}}
```

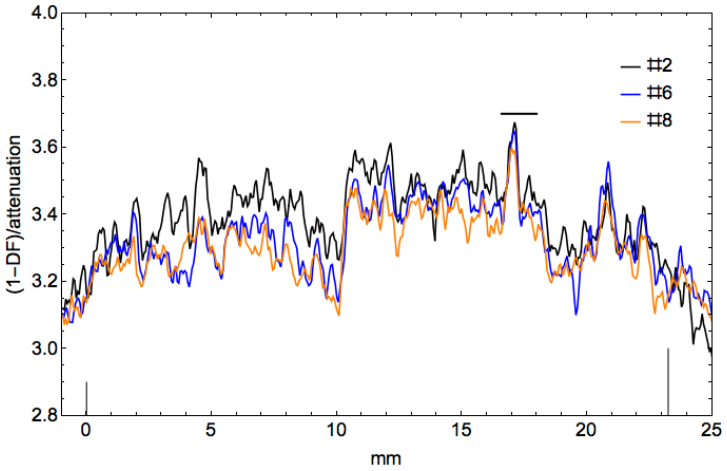
HZB_lineprobe_Expts2-6-8_Step4_right

Fig. **??** shows the code used to make the plot shown in Fig. **??**. The code has features not need in the DragonFly macro: Two vertical markers are drawn to represent the ends of the short and long fractured dogbone. A short horizontal bar is drawn at 17.3±0.7 mm to represent the fracture of the stressed sample post-imaging. The DragonFly macro should draw the trace labeled #2. **Fiducial:** Any ideas on how to make the end of the short fractured sample the 0 mm point on the DragonFly line probe? The Mathematica code uses the slice number of the last intensity of the short fractured sample in the attenuation volume.

```
xAxisExpt2 = -pixelSizeMM * dataBottomExpt2Vertical[[All, 1]] - xMMreference;
xAxisExpt6 = -pixelSizeMM * (-1 * dataBottomExpt6HorV1[[All, 1]] + 885.5) - xMMreference;
xAxisExpt8 = -pixelSizeMM (-1 * dataBottomExpt8HorV2[[All, 1]] + 853) - xMMreference;
ratioDFvAttenExpt2 = dataBottomExpt2Vertical[[All, 2]];
ratioDFvAttenExpt6 = dataBottomExpt6HorV1[[All, 2]];
ratioDFvAttenExpt8 = dataBottomExpt8HorV2[[All, 2]];
gRatioDFvAtten = ListPlot[{Transpose[{xAxisExpt2, ratioDFvAttenExpt2}],
    Transpose[{xAxisExpt6, ratioDFvAttenExpt6}],
    Transpose[{xAxisExpt8, ratioDFvAttenExpt8}] },
   PlotStyle → {Black, Blue, Orange}, Joined → True, Frame → True, FrameStyle → Directive[18, Black],
   FrameLabel → {{"(1-DF)/attenuation", ""}, {"mm", ""}},
   PlotLegends → Placed[{"#2", "#6", "#8"}, Scaled[{0.9, 0.8}]],
   PlotRange → {{-1, 25}, {2.8, 4}}, ImageSize → 700];
gMarkerShortFractured = Graphics[Line[{{xMMshortFractured, 0}, {xMMshortFractured, 2.9}}]];
gMarkerLongFractured = Graphics[Line[{{xMMlongFractured, 0}, {xMMlongFractured, 3.0}}]];
gMarkerHalflifeFractured =
  Graphics[
   {Thick, Line[{{fractureRelativeShortMM - fractureStretchMM / 2, 3.7},
      {fractureRelativeShortMM + fractureStretchMM / 2, 3.7}}]}];
gRatioDFvAtten = Show[{gRatioDFvAtten, gMarkerShortFractured, gMarkerLongFractured, gMarkerHalflifeFractured}]
Export[NotebookDirectory[] <> "Graph_RatioDFvAtten_" <> strNameShort <> ".png", gRatioDFvAtten, "PNG"]
```

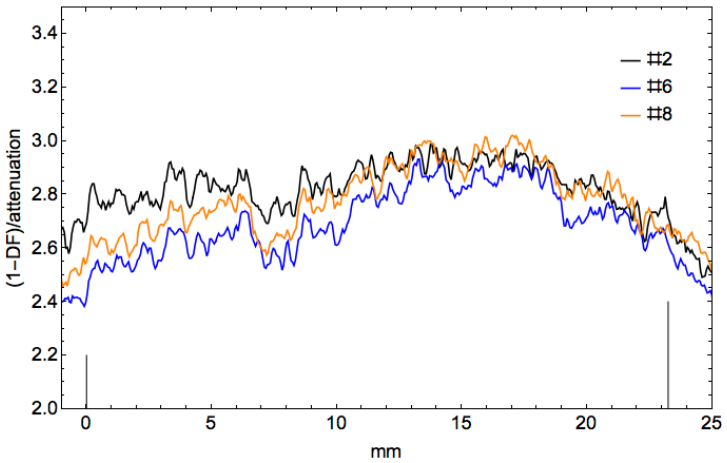
HZB_lineprobe_Expts2-6-8_Step7_1-DF_A

Fig. **??** shows line probe figure. The extremely important aspect of this figure is the match of the short horizontal line at 17.3±0.7 mm with a spike in the trace of (1-DF)/A for all three independent tomography runs. Also, the near overlap of the three traces shows very good reproducibility in the neutron interferometry/tomography; given the low neutron count rate in the experiment and the susceptibility of interferometers to drift, this is surprising.



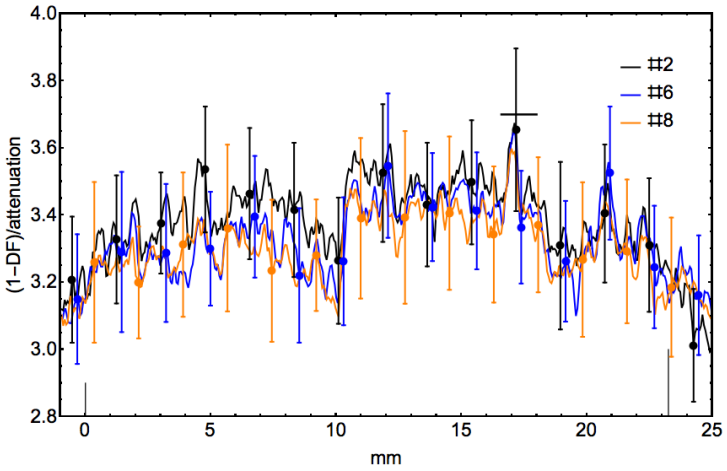
Graph_RatioDFvAtten_erode5

Fig. **??** shows that the pristine sample has print irregularities and suggest heat/pressure annealing is needed for this sample.



Graph_RatioDFvAtten_erode5_pristine

Fig. **??** is the same as Fig. **??**, but with error bars drawn for every 30-th point. The error bars are large because of heterogeneity in the sample and low neutron count per image.



Graph_RatioDFvAtten_erode5_with_errorbars

Appendix E
Matlab Scripts of Harker-O'Leary Algorithm

The main program will be the following.

```
clc; clear;
cd;
pathFigures = cd;

pathGH = '/Volumes/data2/CAMD/peppercorn_March2018/GH/FITS_abs_dpc_DF/';
pathGV = '/Volumes/data2/CAMD/peppercorn_March2018/GV/FITS_abs_dpc_DF/';
%% read fits files
HoriFilename = [pathGH, 'Peppercorn2_GH_dpc-corr_027p500deg.fits'];
VertFilename = [pathGV, 'Peppercorn3_GV_dpc-corr_027p500deg.fits'];

VertData = fitsread(VertFilename);
HoriData = fitsread(HoriFilename);

%%
strFileNamePrefix ='logo_small_';  % micro=5, small=50, large=500 points in membrane
% strFileNamePrefix ='logo_large_'
% micro=5, small=50, large=500 points in membrane
membranePoints=50;
if strFileNamePrefix =='logo_micro_', membranePoints = 5; end;
if strFileNamePrefix =='logo_large_', membranePoints = 500; end;

%ZxN = HoriData(20:230,20:220);
%ZyN = VertData(20:230,20:220);
ZxN = HoriData;
ZyN = VertData;
figure; imshow(ZxN, [-4, 4])
colorbar;
hold on; figure; imshow(ZyN, [-4,4])
colorbar;

%% cropping
%ZxN = ZxN(60:400,60:400);
%ZyN = ZyN(60:400,60:400);
%hold on; figure; imshow(ZxN, [-1, 1])
```

```
%hold on; figure; imshow(ZyN, [-1, 1])
[m, n] = size(ZxN);
x=linspace(1, n, n)';
y=linspace(1, m, m)';
%%  Tikhonov Regularization
if strFileNamePrefix =='logo_micro_',
    N = 3;  % no effect on Matlab logo
    lambda = 0.025 ; % strong effect, best is  0.025 or smaller
else,
    N = 5;  % no effect on Matlab logo
    lambda = 0.001 ; % strong effect, best is  0.025 or smaller
end;
deg = 0 ; % strong effect, best is near 0 (non-negative integers only)
Z0 = 0.1*ones(m,n) ;

%% Expanded diff function for Dx
Dx = dopDiffLocal_simple( x, N, N, 'sparse' ) ;
% full(Dx)

Dy = dopDiffLocal_simple( y, N, N, 'sparse' ) ;
% full(Dy)

%%  Simple g2sTikhonov
tic;
[ Ztik, Res ] = g2sTikhonov_simple( ZxN, ZyN, x, y, N, lambda, deg, Z0, Dx, Dy,
strFileNamePrefix ) ;
timeRequired=toc

%% Plot results
% Ztik = Z;
climZtrue=[0, m, 0, m, -10, 120];
h3 = figure(3);  clf;
s = surface(Ztik);  axis(climZtrue); axis('tight');
    h2 =title(['reconstructed: t=',num2str(timeRequired),' s, \lambda=',
    num2str(lambda)],'FontSize',16);
    xlabel('X'); ylabel('Y');
    set(gca,'FontSize',16);
s.EdgeColor = 'none';
view(3)

%%
%hold on; figure;
imshow(Ztik,[-3,3])

%%
```

```
%hold on; figure;
s = surface(ZyN);  axis(climZtrue);  axis('tight');
    h2 =title('horizontal bunny','FontSize',16);
    xlabel('X'); ylabel('Y');
    set(gca,'FontSize',16);
s.EdgeColor = 'none';
view(3)
print(gcf,[pathFigures,strFileNamePrefix,'reconstructed_original.png'],'-dpng')
```

The purpose of the function scripts is to computes the Global Least Squares reconstruction of a surface.

```
function [ Z, Res ] = g2sTikhonov_simple( Zx, Zy, x, y, N, lambda, deg, Z0,
Dx, Dy, strFileNamePrefix )
%
% Use (syntax):
%   Z = g2sTikhonov( Zx, Zy, x, y, N, lambda, deg )
%   Z = g2sTikhonov( Zx, Zy, x, y, N, lambda, deg, Z0 )
%   [Z,Res] = g2sTikhonov( Zx, Zy, x, y, N, lambda, deg )
%   [Z,Res] = g2sTikhonov( Zx, Zy, x, y, N, lambda, deg, Z0 )
%
% Input Parameters :
%   Zx, Zy := Components of the discrete gradient field
%   x, y := support vectors of nodes of the domain of the gradient
%   N := number of points for derivative formulas (default=3)
%   lambda := either lambda (1x1) or [ lam ; mu ] (2x1), the
%regularization parameter(s)
%   deg := The order of the differential regularization terms (typically
%       0,1, or 2).  Standard form is deg=0.
%   Z0 := (optional) a-priori estimate of the solution surface
%
% Return Parameters :
%   Z := The reconstructed surface
%   Res := Residuals of the LS term and the Regularization term, typically
%       needed for calculating the Regularization parameter (e.g., L-Curve)
%
% Description and algorithms:
%   The algorithm solves the normal equations of the Least Squares cost
%   function with Tikhonov Regularization terms, formulated by matrix algebra:
%   e(Z) = || D_y * Z - Zy ||_F^2 + || Z * Dx' - Zx ||_F^2
%         + lambda^2 * || Dy^(deg) * ( Z - Z0 )
%||_F^2 + mu^2 * || ( Z - Z0 ) * Dx'^(deg) ||_F^2
%   The normal equations are a rank deficient Sylvester equation which is
%   solved by means of Householder reflections and the Bartels-Stewart
%   algorithm.
```

```
%
% References :
%    @inproceedings{
%    Harker2008c,
%       Author = {Harker, M. and O'Leary, P.},
%       Title = {Least Squares Surface Reconstruction from Measured Gradient Fields},
%       BookTitle = {CVPR 2008},
%       Address= {Anchorage, AK},
%       Publisher = {IEEE},
%       Pages = {1-7},
%          Year = {2008} }
%
%    @inproceedings{
%    harker2011,
%       Author = {Harker, M. and O'Leary, P.},
%       Title = {Least Squares Surface Reconstruction from Gradients:
%           \uppercase{D}irect Algebraic Methods with Spectral, \uppercase{T}ikhonov,
%and Constrained Regularization},
%       BookTitle = {IEEE CVPR},
%       Address= {Colorado Springs, CO},
%       Publisher = {IEEE},
%       Pages = {2529--2536},
%          Year = {2011} }
%
% Author :  Matthew Harker and Paul O'Leary
% Date :    17. January 2013
% Version : 1.0
%
% (c) 2013 Matthew Harker and Paul O'Leary,
% Chair of Automation, University of Leoben, Leoben, Austria
% email: office@harkeroleary.org,
% url: www.harkeroleary.org
%
% History:
%   Date:           Comment:
%   Feb. 14, 2011   Original Version
%
    lam = lambda ;
    mu = lambda ;
%=========================================================================
% Generate the Differentiation Matrices and Solve the Sylvester Equation
%=========================================================================
%
% Dx = dopDiffLocal( x, N, N, 'sparse' ) ;
% Dy = dopDiffLocal( y, N, N, 'sparse' ) ;
```

```
% disp('Dx'); full(Dx)
%
tol = sqrt( eps(1) ) ;
%
  %
    A = [ Dy ; mu * speye(length(y)) ] ;
    B = [ Dx ; lam * speye(length(x)) ] ;
    F = [ Zy ; mu * Z0 ] ; % Degree-0 means Dx^0 = I
    G = [ Zx , lam * Z0 ] ;
%=========================
% lyap is in the Matlab Control Systems Toolbox
%=========================
    Z = lyap( full(A'*A), full(B'*B), -A'*F - G*B ) ;

%=========================
% Compute the Residuals:
%=========================
%
    Res = zeros(2,2) ;

    Res(1,1) = norm( Z * Dx' - Zx, 'fro' ) ;
    Res(1,2) = norm( Dy * Z  - Zy, 'fro' ) ;

if strFileNamePrefix =='logo_micro_',
    disp(['A (converted from sparse to full),  ',num2str(size(A))]);
    disp(full(A));
    disp(['B (converted from sparse to full),  ',num2str(size(B))]);
    disp(full(B));
    disp(['F ,  ',num2str(size(F))]);
    disp(F);
    disp(['G ,  ',num2str(size(G))]);
    disp(G);
    disp(['Z ,  ',num2str(size(Z))]);
    disp(Z);
end;

%========
% END
%========

end

function [P,dP, recurrenceCoeffs] = dop_simple( m, n )
%
% Function : Generates a set of discrete orthonormal polynomials, P, and
```

```
%   their derivatives, dP, either of size (m x n), or on the arbitrary
%   support vector x. It also returns the coefficients for the recurrence
%   relationships, these can be used to perform interpolation.
%
% Syntax :
%   [P,dP] = dop( m ) ;
%   [P,dP] = dop( m, n ) ;
%   [P,dP] = dop( x ) ;
%   [P,dP, rC] = dop( x, n ) ;
%
% Input :
%   m := number of evenly (unit) spaced points in support
%   x := arbitrary support vector
%   n := number of functions
%
% Output :
%   P = [ p0, p1, ..., p(n-1) ] := Discete polynomials, pk are vectors.
%   dP = [ dp0, dp1, ..., dp(n-1) ] := Derivatives of the polynomials.
%   rC = a matrix, containing the alpha (col 1) and beta (col 2)
%       coefficients for the three term recurrence relationship
%
% Cite this as :

% @article{DBLP:journals/tim/OLearyH12,
%  author    = {Paul O'Leary and
%               Matthew Harker},
%  title     = {A Framework for the Evaluation of Inclinometer Data in the
%               Measurement of Structures},
%  journal   = {IEEE T. Instrumentation and Measurement},
%  volume    = {61},
%  number    = {5},
%  year      = {2012},
%  pages     = {1237-1251},
% }
%
% @inproceedings{
% olearyHarker2008B,
%   Author = {O'Leary, Paul and Harker, Matthew},
%   Title = {An Algebraic Framework for Discrete Basis Functions in Computer Vision},
%   BookTitle = {IEEE Indian Conference on Computer Vision,
%Graphics and Image Processing},
%   Address= {Bhubaneswar, Dec},
%   Year = {2008} }
%
% Author : Matthew Harker
```

```
% Date : Nov. 29, 2011
% Version : 1.0
%--------------------------------------------------------------------------
% (c) 2011, Harker, O'Leary, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%--------------------------------------------------------------------------
% History:
%   Date:           Comment:
%   Nov. 29, 2011   Original Version 1.0
%--------------------------------------------------------------------------
%
[u,v] = size( m ) ;
%
if u == 1 && v == 1
    %
    x = (-1:2/(m-1):1)' ;
    %
elseif u ~= 1 && v == 1
    %
    x = m ;
    m = length(x) ;
    %
else
    %
    error('Support x should be an m x 1 vector') ;
    %
end
%
if nargin == 1
    n = m ;
end
%
%==============================
% Generate the Basis
%==============================
%
% Generate the first two polynomials :
p0 = ones(m,1)/sqrt(m) ;
meanX = mean( x );
p1 = x - meanX ;
np1 = norm( p1 ) ;
p1 = p1 / np1;
%
% Compute the derivatives of the degree-1 polynomial :
hm = sum( diff( x ) ) ; % Alternatively mean(...)
```

```
h = sum( diff( p1 ) ) ; % Alternatively mean(...), but 1/n cancels.
dp1 = (h/hm) * ones(m,1) ;
%
% Initialize the basis function matrices :
P = zeros(m,n) ;
P(:,1:2) = [ p0, p1 ] ;
%
dP = zeros(m,n) ;
dP(:,2) = dp1 ;
%
% Setup storage for the coefficients of the three term relationship
%
alphas = zeros(n,1);
alphas(1) = 1/sqrt(m);
alphas(2) = 1/np1;
%
betas = zeros(n,1);
betas(2) = meanX;
%
for k = 3:n
    %
    % Augment previous polynomial :
    pt = P(:,k-1) .* p1 ;
    %
    % 3-term recurrence :
    beta0 = (P(:,k-2)'*pt) ;
    pt = pt - P(:,k-2) *  beta0 ;
    betas(k) = beta0;
    %
    % Complete reorthogonalization :
    beta = P(:,1:k-1)' * pt ;
    pt = pt - P(:,1:k-1) * beta ;
    %
    % Apply coefficients to recurrence formulas :
    alpha = 1/sqrt(pt'*pt) ;
    alphas(k) = alpha;
    P(:,k) = alpha * pt ;
    dP(:,k) = alpha * ( dP(:,k-1) .* p1 + P(:,k-1) .* dp1 - dP(:,k-2) *
    beta0 - dP(:,1:k-1)*beta  ) ;
    %
end;
%
recurrenceCoeffs = [alphas, betas];
%========
% END
```

```
%========

function S = dopDiffLocal_simple( x, ls, noBfs, option )
%
% Purpose : This function generates a global matrix operator which
% implements the computation of local differentials where the vector of x
% values may be irregullarly spaced.
%
% In general the support length ls should be an odd number. There is an
% exceltion made upto to ls = 20 and ls = noPoints, in this case a full
% differentiating matrix is computed.
%
% Use (syntax):
%   S = dopDiffLocal( x, ls, noBfs, option )
%
% Input Parameters :
%   x : The vestor of x value fo rthe computation.
%   ls : The support length used for the local differential
%   noBfs : the number of basis functions to be used.
%   option: 'sparse' generated sparse matrix notations, default is full.
%
% Return Parameters :
%   S: The local differential operator
%
% Description and algorithms:
%   Local discrete orthogonal polynomials are used to generate the
%local approximations
%   for the dreivatives
%
%
% Author :  Matthew Harker and Paul O'Leary
% Date :    17. January 2012
% Version : 1.0
%
% (c) 2013 Matthew Harker and Paul O'Leary,
% Chair of Automation, University of Leoben, Leoben, Austria
% email: office@harkeroleary.org,
% url: www.harkeroleary.org
%
% History:
%   Date:           Comment:
%

%-------------------------------------------------------------------------
[noPts, mt] = size(x);
```

```
% Test the input paramaters
%-----------------------------------------------------------------
% Use sparse matrices if necessary
%

    genSparse = true;

    %
    %-------------------------------------------------------------------------
    %
    rows = [];
    cols = [];
    vals = [];
    %
    % Determine the half length of ls this determine the upper ane lower
    % postions of Si.
    %
    ls2 = round( (ls + 1 )/2 );
    %
    % generatethe top of Si
    %
    range = (1:ls)';
    halfRange = (1:ls2)';

    startX = x(range);
    [Gt, dGt] = dop_simple( startX, noBfs );
    Dt = dGt * Gt';
    %
    for k=1:length(halfRange)
        row = halfRange(k) * ones(length(range),1);
        %
        rows = [rows; row];
        cols = [cols; range];
        vals = [vals; Dt(halfRange(k),:)'];
    end;
    %
    % Compute the strip diagonal entries
    %
    noOnDiag = noPts - 2 * ls2;
    for k=1:noOnDiag
        localX = x(range+k);
        [Gt, dGt] = dop_simple( localX, noBfs );
        tdGt = dGt(ls2,:);
        dt = tdGt * Gt';
        row = (k + ls2) * ones( length(range),1 );
```

```
    %
    rows = [rows; row];
    cols = [cols; range + k];
    vals = [vals; dt'];
end;
%
% generate the bottom part of Si
%
endX = x(end-ls+1:end);
[Gt, dGt] = dop_simple( endX, noBfs );
Dt = dGt * Gt';
halfRange = (noPts-ls2+1:noPts)';
range = (noPts-ls+1:noPts)';
for k=1:length(halfRange)
    row = halfRange(k) * ones(length(range),1);
    %
    rows = [rows; row];
    cols = [cols; range];
    vals = [vals; Dt(k+ls2-1,:)'];
end;
%
S = sparse(rows, cols, vals, noPts, noPts );
%
rS = sprank( S );
if rS < noPts - 1
    warning(['The rank of S is ',int2str(rS),'
    while x has n = ',int2str(noPts),' points.']);
end;
```

Vita

Jumao Yuan was born in 1991 in China, Chong Qing. She finished her undergraduate studies in Chemistry Department at Beijing Normal University in May 2012 with a Bachelors of Science. Her undergraduate research focused on the physical polymer fabrication. She joined in Chemistry at LSU from August 2012 and after one year's, she became a member in Dr. Butler's group. Her interests involve Python programming and 3D visualization for interferometry/tomography imaging analysis. During her stay as a candidate of Ph.D. student in Dr. Butler's research group, she developed interferometry analysis and image reconstruction with Jupyter notebook in Python and also collaborated with Michael Marsh from ORS company and Juliana from NYU for 3D visualization. At present, imaging analysis with interferometry/tomography has been implemented from raw images to final 3D visualization. In her future career, she will devote herself to 3D image processing with programming languages.